ON SOME VISIBILITY AND INTERSECTION PROBLEMS IN COMPUTATIONAL GEOMETRY
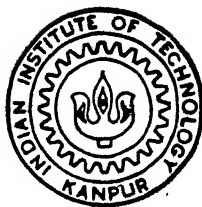
by

G. Srinivasaraghavan

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

August, 1992

ON SOME VISIBILITY AND INTERSECTION PROBLEMS IN COMPUTATIONAL GEOMETRY

A Thesis Submitted

in Partial Fulfilment of the Requirements

for the Degree of

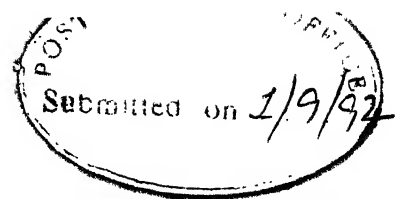Doctor of Philosophy

by

G. Srinivasaraghavan

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
AUGUST, 1992

CERTIFICATE

Submitted on 1/9/92

It is certified that the work contained in this thesis entitled "ON SOME VISIBILITY AND INTERSECTION PROBLEMS IN COMPUTATIONAL GEOMETRY", by G. Srinivasaraghavan, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Thesis Supervisor

Dr. Asish Mukhopadhyay

Department of Computer Science and Engg.

Indian Institute of Technology, Kanpur-208016

India

August, 1992.

SYNOPSIS

This thesis consists of some results on the characterizations of certain kinds of visibility graphs and a result on straight-line stabbing of a set of simple planar objects.

Visibility Graphs

Visibility problems occur often in computational geometry and have been claiming substantial attention from computational geometers for quite some time now. Visibility graphs are combinatorial structures associated with the visibility relationships between pairs of objects making up a geometrical scene. It is felt that a combinatorial study of visibility graphs as objects in their own right is necessary in order that we may gain a deeper understanding of problems involving visibility and thus would help us design better algorithms. Interest in such an investigation has been fairly recent and the available results are very few in number leaving a host of problems to be researched into and solved. The area also promises to be exciting in that the problems that have been thrown up appear rather deep and the techniques that may have to be evolved in order to solve them would certainly enrich our knowledge of the world of computing.

We say that a graph is realizable under a particular kind of visibility if there exists a scene whose visibility graph is the given one. The problem now consists of discovering a set of necessary and sufficient conditions for a given graph to be realizable. A closely related problem is that of designing algorithms to recognize graphs that are realizable. We have studied two kinds of visibility graphs in this thesis—one defined on simple polygons and the other on orthogonal polygons, both *in general position*. We discuss our results on these separately below.

Vertex Visibility Graphs of Simple Polygons

The vertex visibility graph of a simple polygon is defined as the graph consisting of a vertex

ii

for every vertex of the polygon and an edge between two vertices if the line segment joining the two corresponding vertices of the polygon lies entirely within the polygon. The problem in its full generality is still unsolved. The best known bound on the complexity of the recognition problem for these graphs is that it is in PSPACE. Ghosh[1] however proposed a simpler version of the problem where a hamiltonian cycle of the graph is also given as part of the input. It is clear that any such visibility graph must be hamiltonian. The problem now is to recognize/characterize graphs which are realizable as simple polygons with the boundaries corresponding to the given hamiltonian cycles. The complexity of even this problem is unknown. The general problem would be placed in NP if this is solvable in polynomial time. We propose a new necessary condition for this recognition problem. This is essentially a generalization of one of the necessary conditions, relating to the existence of blocking vertices for pairs of invisible vertices, proposed by Ghosh. This new condition was in fact conjectured by Everett[2] and seems to throw more light on the recognition problem for these visibility graphs. This condition insists on the existence of a *consistent* assignment of blocking vertices to invisible pairs of the graph rather than just a blocking vertex for each invisible pair. Our proof of the conjecture is constructive in that given any polygon, our proof tells us how we can make a consistent assignment. The result also looks promising in other ways because from a property of our construction it also follows that there exists an assignment in which a vertex is assigned as a blocking vertex iff it is a concave vertex of the polygon. This property is likely to be extremely useful while attempting a reconstruction of the realizing polygon from a visibility graph.

Orthogonal Edge Visibility Graphs

These graphs introduced by O'Rourke, are defined on orthogonal polygons and capture the visibilities among pairs of edges. We say that a pair of edges of the polygon are visible

---

[1] S.K.Ghosh, *On Recognizing and Characterizing Visibility Graphs of Simple Polygons* in Vol. 318 of LNCS, 1988.

[2] H.Everett, *Visibility Graph Recognition*, Ph.D Thesis, Univ. of Toronto, 1989.

to each other if there exists a rectangle of non-zero area lying wholly within the polygon such that two opposite sides of the rectangle lie on the two edges. The visibility graph of a polygon therefore consists of two (connected[3]) components. In this thesis we give a set of six necessary conditions for a graph to be one of the connected components of the visibility graph of an orthogonal polygon with holes. We also give an example of a graph which satisfies all our conditions but is not realizable, thus making our set of conditions not sufficient. We however show that these conditions are sufficient to obtain a realization of the graph as a polygon one component of whose visibility graph differs from the given graph by at most as many leaves as the number of faces in a planar embedding of the given graph (one of our necessary conditions says that the graph must be planar). Our characterization is complete for trees. We also study the problem of determining if two graphs with the same number of vertices can together be the visibility graph of an orthogonal polygon. For this we give two negative results—we construct two fairly large classes of pairs of graphs for which such a realization is impossible.

Completeness

This is a speculative note on what we believe is a highly desirable property to be satisfied by any algorithm which proposes to reconstruct a scene from a given visibility graph. We illustrate the idea with a simple example.

Given a scene it is clear that the visibility graph of the scene is unique. But there are in general several scenes which have the same visibility graph. In trying to show that some conditions are sufficient for a graph to be a visibility graph, one produces an algorithm to reconstruct *some* scene which realizes an arbitrary graph satisfying the conditions. Such algorithms however throw very little light on the deeper relationships between a scene and its visibility graph. It would be far more desirable to have an algorithm (a generic one) which can potentially produce *every* scene whose visibility graph is the given one. We call

---

[3]O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford Univ. Press, New York, 1987.

such algorithms *complete* reconstruction algorithms. We feel this is necessary for us to get a thorough insight into the combinatorial nature of visibility.

In this part of the thesis, we discuss the completeness criterion for reconstruction algorithms and present an illustrative algorithm which is complete, though in a rather restricted sense. The algorithm is the 'complete version' of a known algorithm due to O'Rourke. We also briefly discuss the implications of this in handling weighted visibility graphs.

Stabbing Problems

The last part of this thesis consists of a solution to a stabbing problem in a fairly general setting. Given a set of planar objects, a stabber for the set is a line which intersects every object in the set. The problem is to determine if a given set of objects admits such a line. The problem has been solved in general only for objects (splinegons[4]) whose boundaries have constant size storage descriptions[5]. We give an algorithm which works for a set of arbitrary splinegons within the same time bounds.

---

[4] Diane L. Souvaine, *Computational Geometry in a Curved World*, Ph.D Thesis, Princeton Univ., 1986.

[5] M. Atallah and C. Bajaj, *Efficient Algorithms for Common Transversals*, IPL 25 (1987), pp. 87–91.

ACKNOWLEDGEMENTS

vi

कायेन वाचा मनसेन्द्रियैर्वा
        बुद्ध्यात्मना वा प्रकृतेः स्वभावात् ।
करोमि यद्यत् सकलं परस्मै
        नारायणायेति समर्पयामि ।।

Contents

List of Tables

List of Figures

xiv

Chapter 1

Introduction

Computational Geometry, now a well established branch of theoretical computer science, deals with the algorithmic and complexity issues of problems with a distinctly geometric flavour. This thesis concerns itself with two classes of geometric problems—visibility problems, particularly those on visibility graphs and intersection problems, in particular 'stabbing problems' in computational geometry. We devote the first two sections of this chapter to a broad introduction to each of the above two classes of problems along with a brief survey of the existing results in each. The last section introduces the notations and the conventions we use throughout this thesis. We conclude this chapter with an overview of the organization of the thesis.

1.1 Visibility Problems

Visibility considerations arise quite naturally in several geometric problems, primarily those arising in robotics and computer graphics. Planning the motion of a robot through a 'workspace' cluttered with 'obstacles' (objects which the robot is not to collide with, during its motion) clearly involves notions of what portions of the workspace can the robot, at any

1

particular moment, 'see' to be able to find its way through. We of course assume that the obstacles are for our purposes 'opaque'. An art-gallery problem, for instance, talks of the portions of a polygonal art-gallery (the sides of the polygon being the walls of the gallery) which a watchman can see from his post. In computer graphics one often deals with the elimination of hidden objects from the display, where the obvious concerns are with what is visible to the observer and what is not. Visibility problems have been studied with several variations of this basic theme. In this thesis we confine our attention to a study of the combinatorial structure of visibility.

Typically a visibility problem is defined on a *scene* which consists of a finite set of *objects* embedded in some space, a *visibility relationship* defined on groups of two or more objects subject to a set of *constraints*. In the rest of the thesis we deal only with scenes in which the visibility relationship, not necessarily symmetric, is defined only on pairs of objects. Indeed this is the simplest case and is the only one to have been investigated in the literature on the subject. We say that an object *A is visible to* or *can see* another object *B* if *A* is related to *B*. Moreover notions of visibility between pairs of objects are invariably defined in terms of a more fundamental, and often implicit notion of visibility between a pair of points of the space in which the scene is embedded. Typically, two points are said to be visible to each other if there exists a path (called the *visibility path*) between the two points such that it satisfies the visibility constraints. As mentioned earlier, the objects of the scene are considered 'opaque' and thus the constraints specify that a visibility path cannot intersect any object. However, 'grazing' intersections are normally allowed. The other visibility constraints are usually of the following kinds:

- A visibility path must lie within a given region of the underlying space.

- A visibility graph must intersect none of a given set of 'obstacles'. Again 'grazing' intersections are usually allowed.

The class of visibility paths allowed determine the 'notion of visibility'. Note that the

point-visibility relation is symmetric. The interpretation of the visibility between a pair of 'macro' objects A and B in terms of the visibilities between pairs of points can essentially be of the following four kinds:

1. *Total visibility:* A and B are said to be totally visible to each other if for any pair of points $x \in A$ and $y \in B$, x can see y. This is a symmetric relation.

2. *Strong visibility:* B is said to be strongly visible from A if there exists a point $x \in A$ such that every point in B is visible to x. This is an asymmetric relation.

3. *Weak visibility:* B is said to be weakly visible from A if for every point $y \in B$ there exists a point $x \in A$ such that x sees y. This too is an asymmetric relation.

4. *Partial visibility:* A and B are said to be partially visible to each other if there exist points $x \in A$ and $y \in B$ such that x can see y. This is a symmetric relation.

Some of these have been defined and studied in specific contexts [5, 24, 10, 41].

Finally, a more general 'theory' of visibility could conceivably try to incorporate the relevant metric information connected with a scene. For instance, the *distance* between a pair of objects i.e., a measure of 'how far' one object is from the other under a given metric, could be one such. Henceforth, we use the term *scene* to mean a scene under the notion of visibility and metric, which is clear from the context, and where none exists, to mean a scene under some *given* notion.

1.1.1   Visibility graphs

The *visibility graph* of a scene is a graph with a vertex for every object of the scene, and an edge between two objects if the two are visible to each other. Note that for a more general notion of visibility, the corresponding representation could turn out to be a hypergraph (for ternary and other higher order relations) and a directed graph if the relation is asymmetric. Under a suitable *metric-interpretation* one can also define a *weighted*

visibility graph consisting of weights for each of its edges. A graph is said to be *realizable* (under some given notion of visibility) if there exists a scene whose visibility graph is the given graph, and such a scene is said to be a *realization* of the given graph.

There have been two main directions in the study of visibility graphs—one algorithmic in nature and the other graph theoretic. In the first, one wants to compute the visibility graph of a given scene efficiently. In the second, one studies graphs which are realizable, as combinatorial objects in their own right. One looks for properties which characterize such graphs, algorithms that can identify such a graph efficiently, algorithms that can reconstruct a scene which is a realization of a given graph etc.. One may also be interested in seeing if problems known to be intractable for graphs in general, are tractable when restricted to visibility graphs. This last kind of investigation may lead to solutions of some partition or covering problems as shown by Motwani et. al. [34, 35]. Research in all these is still in its infancy, and very few results are available as of today.

Among the former class of problems, investigations in the context of *Euclidean visibility*—the visibility path between a pair of points under this notion is simply the straight line segment joining the two points in the underlying euclidean space—has received almost all the attention. The strongest motivation for construction of visibility graphs for this is its application to the shortest path problem. Among the notable visibility graph construction results are the following.

1. Construction of the vertex visibility graph of a set of line segments in the plane: The vertex visibility graph of a set of line segments is defined on a scene consisting of a set of line segments as obstacles, and the objects of visibility being the end-points of the segments. Welzl [42] gave a worst-case optimal $O(n^2)$ algorithm. Later Ghosh and Mount [21] gave an optimal output-sensitive algorithm to compute this visibility graph. Incidentally, it has been shown by Shen and Edelsbrunner [39] that $5n - 4$ is a tight lower bound on the number of edges of such a visibility graph.

2. Construction of the vertex visibility graph of a simple polygon: Here the scene consists

of a simple polygon in which a pair of vertices of the polygon are said to be visible to each other if the segment joining the two lies wholly within the polygon. Hershberger [28] gave an output-sensitive $O(h + T)$ algorithm to compute the visibility graph of a simple polygon, h being the size of the visibility graph, and T the time taken to triangulate a simple polygon. In the light of the recent linear time triangulation algorithm of Chazelle [9], Hershberger's algorithm becomes optimal.

In this thesis we concern ourselves only with problems in the latter class. It is not clear what immediate application (apart from a few) such a study might have, but it is believed that "...some of the fundamental unsolved problems involving visibility in computational geometry will not be solved until the combinatorial structure of visibility is more fully understood"[1]. It is in this spirit, that we take to this study. We broadly survey some of the existing results below. We give a more elaborate description of the existing results on the two classes of visibility graphs we study in this thesis viz., vertex visibility graphs of simple polygons and orthogonal edge visibility graphs, in the respective chapters.

Vertex visibility graphs of simple polygons happen to be the most widely studied among visibility graphs in general. Everett [19] established a bound on the complexity of the recognition problem for these graphs. She showed that the problem is in PSPACE by expressing the decision problem of realizability as a sentence in the existential theory of reals. Canny [8] had earlier shown that deciding the truth of a sentence in the existential theory of reals is in PSPACE. This is the best known complexity bound for this recognition problem. Everett [19] has also shown that there exists an infinite family of minimal forbidden induced subgraphs for these visibility graphs thus showing that a forbidden subgraph characterization, similar to the one for planar graphs by Kuratowski, does not exist for these visibility graphs.

Orthogonal edge visibility graphs were introduced by O'Rourke [36] and all the earlier results on these are due to him. Wismath [44] studied the vertical visibility between

---

[1]O'Rourke in *Art gallery Theorems and Algorithms*, Oxford Univ. Press, New York, 1987.

a set of X-intervals obaining a complete characterization of the associated visibility graphs using a variation of the *st*-numbering of planar graphs. These graphs were also studied independently by Luccio, Mazzone and Wong [32]. Later Kirkpatrick and Wismath [31] also gave a solution to a weighted version of the problem where the weights were interpreted as the existence of vertical visibility 'beams' of width equal to the weights. Motwani et. al., [34, 35] showed that certain kinds of visibility graphs defined on orthogonal polygons are perfect graphs. The interesting fact about these visibility graphs is that problems like covering an orthogonal polygon with the smallest number of orthogonally star or convex polygons can be translated into an equivalent graph-theoretic problem like clique-cover, on the associated visibility graph. The fact that these graphs are perfect for many of the cases, makes the corresponding polygon decomposition problems polynomial time solvable.

Among instances of attempts at trying out well known graph problems on visibility graphs are the following two results on the vertex visibility graphs of polygons with holes, both due to Everett [19]:

- The Maximum clique problem on vertex visibility graphs is NP-complete.

- The graph isomorphism problem on these visibility graphs is isomorphism-complete.

1.1.2 'Complete' reconstruction algorithms

This is a speculative note on what we believe is a highly desirable property to be satisfied by any algorithm which proposes to reconstruct a scene from a given visibility graph.

Given a scene it is clear that the visibility graph of the scene is unique. But there are, in general, several scenes which have the same visibility graph. In trying to show that some conditions are sufficient for a graph to be a visibility graph, one produces an algorithm to reconstruct *a* scene which realizes an arbitrary graph satisfying the conditions. Such algorithms however throw very little light on the deeper relationships between a scene and its visibility graph. It would be far more desirable to have an algorithm (a generic one)

6

which can potentially produce *every* scene whose visibility graph is the given one. We call such algorithms *complete* reconstruction algorithms. We feel this is necessary for us to get a thorough insight into the combinatorial nature of visibility. We discuss this in chapter 5 of the thesis.

1.2 Stabbing Problems

A *stabbing line* (also known as a *stabber* or a *line transversal*) for a collection of objects in the plane is a line which intersects every object in the set. The problem of computing a line transversal for a given set of objects in the plane or a representation of all the possible line transversals for the set of objects has received considerable attention among computational geometers of late, again motivated for a large part from problems occuring in robotics. Apart from being an interesting problem in its own right—there is a substantial body of 'pure-mathematical' literature too on the line transversal problem [22, 25, 12]—even algorithmically it has necessitated the use of many interesting and wide-ranging techniques like linear programming in fixed dimensions [33, 14], geometric transformations, particularly dualization [23, 7] and recently, techniques connected with Davenport-Schinzel sequences [13, 2, 38] and the computation of lower (upper) envelopes of functions [29]. A variety of algorithms have been proposed for solving special cases of this problem involving objects of specific kinds like, a set of line segments [17], convex polygons [16], equal radius circles [6], circles [30], translates of an object [15], homothets [15] etc.. All these algorithms run optimally in time $O(n \log n)$, in light of the proof of a corresponding lower bound by Avis, Robert and Wenger [4] for a family of n line segments or a family of n circles. The most general algorithm for the line transversal problem known to date is the one by Atallah and Bajaj [3] which can handle sets consisting of objects which are simple planar point sets whose boundaries have a constant size storage description. Their algorithm runs in time $O(\lambda(n) \log n)$ where $\lambda(n)$ is an almost linear function of n. In this thesis we give an algorithm to solve the line transversal problem in a fairly general setting. Our algorithm

works for arbitrary splinegons [40] and matches the time bounds obtained by Atallah and Bajaj.

1.3 Notations, Conventions and Definitions

We use an orthogonal frame of reference throughout, with an X-axis (or the horizontal axis) and a Y-axis (or the vertical axis). We say that a segment is horizontal (vertical) if it is parallel to the X-axis (Y-axis). We frequently use the terms 'up' or 'above', 'down' or 'below', 'left' and 'right' to mean the directions positive Y, negative Y, negative and positive X respectively.

A line through points x and y is denoted as $\overleftrightarrow{xy}$, the line $\overleftrightarrow{xy}$ directed from $x(y)$ to $y(x)$ as $\overrightarrow{xy}$ ($\overleftarrow{xy}$) and the line segment joining the two as $\overline{xy}$. We use the notation (x, y) to denote both the open set $\overline{xy} - \{x, y\}$ and just a pair of points. The intended meaning of (x, y) will be clear from the context. A set of real values A is called an *interval* or a *range* if $a, b \in A$ are any two numbers in A, with $a < b$, then for any other number c between a and b, $c \in A$. We denote an interval bounded by a and b as $[a, b]$, $(a, b]$, $[a, b)$ or (a, b), where '[' and ']' denote that the interval is closed and, '(' and ')', that it is open, at the appropriate end. Throughout this thesis we normally consider only directed lines unless otherwise mentioned. For a point x on a line L, $(x^+)_L$ and $(x^-)_L$ denote respectively the two components of $L - x$ directed *away from* and *into* x, while $[x^\pm]_L = (x^\pm)_L \cup \{x\}$. For a pair of points x and y on L we say that x *precedes* or *is before* y if $y \in (x^+)_L$ and that it *follows* or *is after* y if $x \in (y^+)_L$. The subscript L is omitted if the line being referred to is understood.

Given an oriented, continuous, simple (non self-intersecting) curve C, for any two points x and y on C, $C[x, y]$ ($C(x, y)$) denotes the part of C from x to y inclusive (exclusive) of both, if it exists, in the direction of orientation. If C is piecewise linear then it is called a *polygonal chain* or simply a *chain* with the straight line segments of C called the *edges*

and the "bends", the *vertices*. If C is closed i.e., a Jordan curve, then ΨC and ΓC denote respectively the interior and the closure (i.e., $C \cup \Psi C$) of the region bounded by C. A *polygon* P is a closed polygonal chain, usually represented by the sequence $v_1, v_2, \ldots, v_n$ of its vertices with $P = \cup_{i=1}^{n-1} \overline{v_i v_{i+1}}$ (index arithmetic is modulo n). We also usually make a non-degeneracy assumption that no two consecutive edges of the polygon are collinear. For convenience, we refer to the interior of the region bounded by a polygon as the interior of the polygon. Moreover, as a convention we take all our polygons, unless otherwise mentioned, to be directed clockwise with the interior lying on the right of the polygon during a directed traversal. A vertex v_i of a polygon P is said to be *convex* if for θ_i, the angle from $\overline{v_{i-1} v_i}$ to $\overline{v_i v_{i+1}}$ in the counterclockwise direction (i.e., through ΨP), $\theta_i < \pi$ and *concave* if $\theta_i > \pi$. A polygon P with *holes* is a collection of simple polygons $P = \cup_{i=0}^{h} P_i$ such that $\cup_{i=1}^{h} \Gamma P_i \subset \Psi P_0$ and $\Gamma P_i \cap \Gamma P_j = \phi$, $1 \le i \ne j \le h$. P_0 is called the *outer boundary* of P and the P_i's, $1 \le i \le h$ are called the *holes*. In this case we define ΨP as $\Psi P_0 - \cup_{i=1}^{h} \Gamma P_i$ and ΓP is defined as before.

We say that a polygon (with or without holes) is in *general position* if no three of its vertices are collinear. We deal only with polygons in general position throughout this thesis and the term polygon will refer to those in general position.

· An Orthogonal polygon is a polygon each of whose edges is either horizontal or vertical. Note that on an orthogonal polygon the vertical and horizontal edges alternate. An orthogonal polygon with holes is one whose outer boundary as well as the holes are all orthogonal polygons. We say that an orthogonal polygon is in general position if no two vertices of the polygon can be joined by a horizontal or a vertical segment lying wholly in its interior.

A graph $G = (V, E)$ consists of a set of *nodes* or *vertices* $V(G)$ and a set of *edges* $E(G) \subseteq V(G) \times V(G)$. Two nodes u and v of G are said to be adjacent if $(u, v) \in E(G)$. We denote an edge between a pair of nodes u and v as (u, v). Since we refer to graphs only in the context of visibility, we say that a pair of vertices (u, v) is a visible (invisible) pair if

9

$(u, v) \in E(G) (\notin E(G))$. The rest of the terminology we use in connection with graphs is as in Harary [26]. A visibility graph of a polygon P describes the visibility relationship under some notion of visibility between pairs of elements (vertices, edges etc.) of P with a node for each such element of P and an edge between a pair of nodes if the two corresponding elements are visible to each other in the chosen notion of visibility. If G is the visibility graph of a polygon P then we denote the element of P corresponding to a node v of G as v^* and vice versa. A polygon P is said to *realize* a graph G if G is the visibility graph of P (under the given notion of visibility).

Table 1.1 shows our notational conventions, in which the right hand column shows the symbols normally used to denote the objects of the kind listed in the left hand column. For two objects (lines, polygonal chains etc.) X and Y where X is oriented we use first$(X \cap Y)$ and last$(X \cap Y)$ to denote respectively the first and the last points of intersection between X and Y encountered during a directed traversal of X. We of course require that X and Y intersect at most finitely many times. Finally, throughout the thesis we use the symbol 'ℐ' to denote the end of the proof of a lemma or a theorem.

Finally a note regarding the figures in this thesis. The polygon edges are usually denoted by *thick* lines in the figures, sometimes with an arrow near one of the ends pointing to the side of the edge on which the interior of the polygon lies.

1.4   Thesis Overview

The chapters 2–5 of the thesis deal with visibility graphs. Chapter 2 is on vertex visibility graphs of simple polygons in which we have settled a conjecture by Everett [19] and thus have provided a new necessary condition for a restricted version of the recognition problem for these graphs. In Chapters 3–4 we study orthogonal edge visibility graphs preceded by an introduction to these. We study the realizability of a single connected graph as an orthogonal polygon with holes, one component of the visibility graph of which is the given

Item	Notation
edges, vertices, nodes, points etc.	lower-case a–h, p–q and u–w
integers and indices	lower-case i–n and r–t
coordinate axes	X and Y
ranges and intervals	I, R
graphs	G–H
'macro' objects like polygons, chains, lines etc.	other upper-case alphabets
coordinates and miscellaneous	x, y and z
left and right sides/end-points etc.	$l()$ and $r()$ respectively
left and right regions	$\mathcal{L}$ and $\mathcal{R}$ respectively
two-dimensional regions (typically unbounded)	$\mathcal{A}$, $\mathcal{B}$, etc.
object corres. to node v of the visib. graph	v^*
time-complexity etc.	T

Table 1.1: The notational conventions used in the thesis. Incidentally from a symbol x, other symbols like x', x'', x_1, x_2 etc., may be generated to denote objects of the same class as x.

graph. We give a set of six necessary conditions for a graph to be thus realizable. We end the chapter with a partial sufficiency result. In Chapter 4 we construct two fairly large classes of pairs of trees which are not jointly realizable as orthogonal polygons without holes. In chapter 5 of the thesis, we discuss the completeness criterion for reconstruction algorithms and present an example of an algorithm which is complete. Our algorithm is the 'complete version' of a known algorithm. We also briefly discuss the implications of this in handling weighted visibility graphs. Finally in chapter 6 we develop an algorithm to compute a complete representation of all the possible stabbers of a set of simple splinegons in the plane. We end the thesis with some concluding remarks and suggestions for future work in chapter 7.

Chapter 2

Vertex Visibility Graphs of Simple Polygons

In this chapter, we study what are known as *vertex visibility graphs of simple polygons* which happen to be the most widely investigated in the literature among visibility graphs in general. Here, we also give a new necessary condition for a graph to be the visibility graph of a simple polygon by proving a conjecture by Everett [19].

Given a simple polygon $P = v_1^*, v_2^*, \ldots, v_n^*$ on n vertices, the vertex visibility graph $G = (V, E)$ of P consists of n vertices $v_1, v_2, \ldots, v_n$, v_i corresponding to v_i^* for $1 \leq i \leq n$ and $(v_i, v_j) \in E$ if and only if $\overline{v_i^* v_j^*} \subset \Gamma P$.

In the first section of this chapter, we review some of the earlier work on the recognition problem for these visibility graphs. In section 2 we state Everett's conjecture and prove it. We end the section with some observations on the implications this might have towards a complete characterization of these visibility graphs.

2.1 Earlier Work

In spite of several attempts at a characterization of these graphs, the problem in its full generality still remains unsolved. Partial results however exist.

The following are the two known necessary conditions for G to be a visibility graph, the first one following from the rather obvious fact that the cycle of G corresponding to the boundary of the polygon is a hamiltonian cycle.

Necessary Condition S1: *(Ghosh [22]) G must be hamiltonian.*

Necessary Condition S2: *(Coullard and Lubiw [11]) Let $v_1v_2v_3$ be any three vertices of a triconnected component G' of G, such that v_1, v_2 and v_3 form a triangle. Then G' must have a vertex ordering starting from v_i, $1 \leq i \leq 3$, in which every vertex is adjacent to a 3-clique induced on vertices before it in the ordering.*

Incidentally, the above condition S2 was used by Coullard and Lubiw [11] to give a polynomial time algorithm to determine if G *with weights for each of its edges* is the visibility graph of a simple polygon with the given weights as the Euclidean distances between the corresponding pairs of vertices of the polygon.

The other partial results fall essentially under three categories:

1. Identifying classes of graphs which are visibility graphs of simple polygons.

2. Attempting characterizations of the visibility graphs of some restricted subclass of the class of simple polygons.

3. Attempting characterizations of visibility graphs whose relationships to the corresponding polygons are in some sense 'stronger' than just being a visibility graph of the polygon.

The most obvious fact under the first category is that every complete graph is a visibility

graph—the visibility graph of a polygon is complete if and only if the polygon is convex. Trivially, not every visibility graph is complete. The only other (non-trivial) result of this kind known till date is that every *maximal outerplanar graph* is the visibility graph of a monotone polygon (ElGindy [18]). It is however easy to verify that in fact maximal outerplanarity completely characterizes the visibility graphs of those polygons which have a *unique triangulation*. It is incidental that some uniquely triangulable polygons are also monotone.

The only known results in the second category are the ones for the visibility graphs of spiral polygons by Everett and Corneil [20] and for the visibility graphs of staircase polygons by Abello, Egecioglu and Kumar [1]. A *k*-spiral polygon is one which has at most one maximal contiguous sequence of concave vertices on its boundary. Everett and Corneil [20] have shown that the class of visibility graphs of spiral polygons form a subclass of chordal graphs and are completely characterized with an extra condition. Everett [19] has also shown that the visibility graphs of 2-spiral polygons are perfect graphs. A staircase polygon is an orthogonal polygon one vertex of which is visible to every other vertex of the polygon. Abello et. al. show the necessity and sufficiency of a 'persistence' condition on the adjacency matrix of the given graph for it to be the visibility graph of a staircase polygon. As for polygons with holes, Everett [19] has shown that the visibility graphs of convex polygons with convex holes is a circular-arc graph.

The only known attempt in the third category is the one by Ghosh [22]. Starting from the fact that a visibility graph is necessarily hamiltonian, Ghosh considered the problem of trying to determine if the given graph *along with a hamiltonian cycle on it* is the visibility graph of a simple polygon whose boundary corresponds to the given hamiltonian cycle. He proposed a set of necessary conditions for this and conjectured that these were sufficient as well. Everett [19, section 2.2.3] later showed that these set of conditions were in fact not sufficient.

As we mentioned earlier, PSPACE happens to be the best known bound on the

15

complexity of the recognition problem. It would however be placed in NP if a polynomial time solution to Ghosh's version of the problem is found. Our result in this chapter makes a step towards this. We henceforth, deal only with the above mentioned Ghosh's version of the visibility graph recognition problem.

Consider a graph $G = (V, E)$ and a hamiltonian cycle $H = v_1, v_2, \ldots, v_n, v_1$, $v_i \in V$ on it. A cycle C of G is said to be *ordered* with respect to H if C is $v_{i_1}, v_{i_2}, \ldots, v_{i_k}, v_{i_1}$ for some $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ and is said to be *chordless* if the only edges in the subgraph of G induced on the vertices v_{i_j}, $1 \leq j \leq k$, are the cycle edges (v_{i_1}, v_{i_k}) and $(v_{i_j}, v_{i_{j+1}})$, $1 \leq j < k$. We can now state the first of the necessary conditions of Ghosh.

Necessary Condition S3: *[22] G has no ordered chordless cycle of length at least four.* ·

For any pair of vertices (v_i, v_j), $i \neq j$, chain(v_i, v_j) is defined as the ordered set $\{v_i, v_{i+1}, v_{i+2}, \ldots, v_j\}$ of vertices where all index summations are modulo n. A vertex $b = v_i \in$ chain$(u, v) - \{u, v\}$ is called a *blocking vertex* of the pair (u, v) if for any two vertices $p \in$ chain(u, v_{i-1}) and $q \in$ chain(v_{i+1}, v), (p, q) is an invisible pair. An invisible pair is said to be *minimal with respect to b* if b is its only blocking vertex. Two invisible pairs (u, v) and (p, q) are said to be *separable with respect to $b = v_i$* if p and q are encountered before u and v (or vice versa) in chain(v_i, v_{i-1}). We will now state the other two necessary conditions of Ghosh.

Necessary Condition S4: *[22] Every invisible pair in G has a blocking vertex.*

Necessary Condition S5: *[22] If two invisible pairs are separable with respect to vertex b then they cannot both be minimal with respect to b.*

Everett [19] also conjectured a stronger version of condition S5 after proving that the above conditions given by Ghosh were not sufficient. We devote the rest of this chapter to the proof of this conjecture.

16

2.2 Everett's Conjecture

Everett [19] conjectured the following stronger version of Necessary Condition S5:

Theorem 2.1: *Given the graph $G = (V, E)$ with the hamiltonian cycle H of G, if G is the visibility graph of a simple polygon with H corresponding to the boundary of the polygon then, there exists an assignment $B : (u, v) \mapsto b$ of blocking vertices to invisible pairs such that for any two distinct invisible pairs (u, v) and (p, q), $B((u, v)) = B((p, q)) = b$ implies (u, v) and (p, q) are not separable with respect to b.*

An assignment of blocking vertices to invisible pairs satisfying the conditions of the above theorem is called a *consistent* assignment. Note that both the necessary conditions S4 and S5 follow from Theorem 2.1. We now prove the theorem below by first specifying an assignment for the visibility graph of an arbitrary simple polygon P, through Lemmas 2.2.1–2.2.3 and later proving its consistency. We henceforth work only in the context of a polygon throughout our proofs. Thus to avoid clumsiness we denote the polygon vertices too by the corresponding 'unstarred' names of the respective visibility graph vertices.

We first need some more definitions here. A chain $P[p, q]$ of a polygon P is called a *forward chain* (abbreviated $\mathcal{F}$-chain) with respect to a line L if $p, q \in L$ with p preceding q on L and $P(p, q)$ lies wholly on one side of L. $P[p, q]$ would be analogously called a *backward chain* ($\mathcal{B}$-chain) if instead p *follows* q. We use $\mathcal{L}$ and $\mathcal{R}$ to denote the $\mathcal{B}$- or $\mathcal{F}$-chains lying wholly on the left and the right of L respectively. An $\mathcal{LF}$-chain for instance, is a forward chain on the left of L. We say that an $\mathcal{R}$- or $\mathcal{L}$-chain $P[p, q]$ *encloses* a segment (point) $\overline{xy}$ (x) if $\overline{xy} \subset \overline{pq}$ $(x \in (p, q))$. The *span* of such a chain, denoted as span($P[p, q]$), is the interior of the region bounded by $P[p, q] \cup \overline{pq}$.

For a pair of vertices (u, v) of P, consider say the chain $P[u, v]$. Let $P[x, y]$ and $P[x', y']$ both subsets of $P[u, v]$, be two $\mathcal{R}$-chains with respect to $\overrightarrow{uv}$. Since P is simple, clearly either $\overline{xy} \cap \overline{x'y'} = \phi$ or one of the two is contained in the span of the other. We can

17
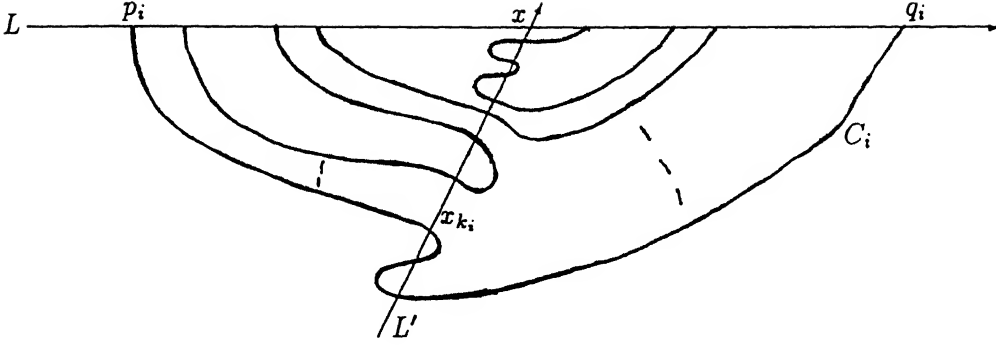
Figure 2.1: The $\mathcal{R}$-chains enclosing a point x on a line L which intersects P

thus define a partial order '$\prec$' on these chains such that $P[x,y] \prec P[x',y']$ if and only if $P[x,y] \subset \text{span}(P[x',y'])$. A chain C' would be called an *immediate superior* of C if $C \prec C'$ and for no other chain C'' is $C \prec C'' \prec C'$. A chain C is called the *highest* (*lowest*) with respect to $\prec$ if for no other chain C' is $C \prec C'$ ($C' \prec C$). An $\mathcal{RF}$-chain with respect to $\overrightarrow{uv}$ is called a *blocking chain* of (u,v) in $P[u,v]$ if its immediate superior in the ordering $\prec$ is not an $\mathcal{RB}$-chain of $P[u,v]$ with respect to $\overrightarrow{uv}$. Henceforth whenever we refer to an $\mathcal{R}$- or an $\mathcal{L}$-chain of $P[u,v]$, we mean a subchain of $P[u,v]$ which is an $\mathcal{R}$- or $\mathcal{L}$-chain with respect to $\overrightarrow{uv}$. Analogous definitions hold for $\mathcal{B}$- and $\mathcal{F}$-chains and also for the subchains of $P[v,u]$. We will now prove a fact about the arrangement of the $\mathcal{R}$- and $\mathcal{L}$-chains of P with respect to an arbitrary line L. We will use this fact later in our proofs.

Lemma 2.2.1: *Let L be a directed line intersecting P and let $x \in L - P$ be a point on it. If $C_1 \prec C_2 \prec C_3 \prec \ldots \prec C_t$ are the $\mathcal{R}$-chains of P with respect to L which enclose x, then for $x \in \Psi P$ ($x \notin \Psi P$) C_i is an $\mathcal{RB}$-chain ($\mathcal{RF}$-chain) for all odd i and an $\mathcal{RF}$-chain ($\mathcal{RB}$-chain) for every even i. Also t is odd if $x \in \Psi P$ and even otherwise. A similar statement holds for the $\mathcal{L}$-chains of P as well.*

Proof: Let L' be another line through x not coincident with L and directed to its left. See Figure 2.1. Consider an $\mathcal{R}$-chain C of P with respect to L. If C encloses x, since the two endpoints are on either side of L' the number of intersections of C with $(x^-)_{L'}$ (we will

18

denote this henceforth as $(x^-)'$ and $(x^\pm)_L$ as simply $(x^\pm)$) is odd. Note that $C \cap (x^+)' = \phi$. Similarly if C does not enclose x then $|C \cap (x^-)'|$ is even.

Suppose $x \in \Psi P$. If $t = 0$ i.e., if there are no $\mathcal{R}$-chains of P enclosing x then clearly $|P \cap (x^-)'|$ is even. Let $P \cap (x^-)' = \{x_1, x_2, \ldots, x_m\}$ where x_i precedes x_{i-1} on L' for every $i > 1$. Because $x \in \Psi P$ and $(x, x_1) \cap P = \phi$, we have $(x, x_1) \subset \Psi P$ implying in turn that $(x_i, x_{i+1}) \subset \Psi P$ whenever i is even (this follows from the easily verifiable fact that the segments formed by the intersections of a line with a polygon are alternately inside and outside the region enclosed by the polygon). Therefore, m being even, it must be that $(x_m^-)' \subset \Psi P$. But this cannot happen because ΨP is bounded. Thus we assume henceforth that $t \geq 1$ (we have nothing to prove if $t = 0$ for $x \notin \Psi P$).

Let x_{k_i} be the point of intersection of C_i with $(x^-)'$ closest to x and let $C_i \cap (x^-) = p_i$ and $C_i \cap (x^+) = q_i$ for some i, $1 \leq i \leq t$. See Figure 2.1. We will first count the number of intersections of the $\mathcal{R}$-chains of P with (x, x_{k_i}). Consider an $\mathcal{R}$-chain $C \subset \operatorname{span}(C_i)$ (note that because P is simple, only such chains can intersect (x, x_{k_i})) and let Q be the polygon $\overline{xq_i} \cup \overline{xx_{k_i}} \cup$(the part of C_i between x_{k_i} and q_i). We now use the fact that for any Jordan curve J and any other open continuous curve S, both of whose end points lie outside(inside) the region bounded by J, $|S \cap J|$ is even (or zero). Also if one of the endpoints is inside and the other outside then $|S \cap J|$ is odd.

If C encloses x then let $p = C \cap (p_i, x)$ and q a point on C very close to $C \cap (x, q_i)$. Consider the portion C' of C between p and q. From our observation above, it follows that $|C' \cap Q|$ is odd. But $C' \cap \overline{xq_i} = C' \cap$(part of C_i between x_{k_i} and q_i)$= \phi$. Thus $|C \cap \overline{xx_{k_i}}| = |C' \cap \overline{xx_{k_i}}|$ is odd. However if C does not enclose x (then either both its end points are in (p_i, x) or both are in (x, q_i)), it should be clear, from a similar argument, that $|C \cap \overline{xx_{k_i}}|$ is even. Hence since there are $(i - 1)$ $\mathcal{R}$-chains enclosing x in the span of C_i, clearly $|(x, x_{k_i}) \cap P| = k_i - 1$ is even if and only if i is odd. Now if $x \in \Psi P$ ($x \notin \Psi P$), we have $(x, x_1) \subset \Psi P$ ($(x, x_1) \not\subset \Psi P$), implying that $(x_{k_i-1}, x_{k_i}) \subset \Psi P$ if and only if $(k_i - 1)$ is even(odd). Therefore since P is directed clockwise, C_i crosses L' at x_{k_i} from the right of

L' to its left if and only if i is odd (even).

To complete the proof, we only have to show that if C_i crosses L' at x_{k_i} from the right(left) to its left(right) then C_i is an $\mathcal{RB}$-chain($\mathcal{RF}$-chain) of P with respect to L.

Suppose on the contrary, C_i crosses L' at x_{k_i} from the left of L' to its right and is an $\mathcal{RB}$-chain. Since x_{k_i} is the point in $C_i \cap (x^-)'$ which is closest to x, we have $P(q_i, x_{k_i}) \cap (x, x_{k_i}) = \phi$. Thus $Q = \overline{xx_{k_i}} \cup \overline{xq_i} \cup P[q_i, x_{k_i}]$ is a simple polygon with C_i directed *into* ΨQ at x_{k_i}. Also $p \notin \Psi Q$. Hence $P(x_{k_i}, p_i) \cap (x, x_{k_i}) \neq \phi$ (since $P(x_{k_i}, p_i) \cap Q \neq \phi$ and $P(x_{k_i}, p_i) \cap P(q_i, x_{k_i}) = P(x_{k_i}, p_i) \cap (x, q_i) = \phi$). But then x_{k_i} cannot be the point of $C_i \cap (x^-)'$ closest to x giving rise to a contradiction. The case where C_i crosses L' at x_{k_i} from the right to the left is analogous (we take the polygon $P[p_i, x_{k_i}] \cup \overline{xx_{k_i}} \cup \overline{p_i x}$ and then arrive at a contradiction just as above). Hence the lemma. ∎

Our definition of a blocking chain is motivated by the fact that every invisible pair has at least one reflex blocking vertex lying on a blocking chain of the pair. We will eventually define B to assign only such blocking vertices to invisible pairs. But first we confirm that every invisible pair does indeed have a blocking chain and later show the existence of such blocking vertices through the following two lemmas.

Lemma 2.2.2: *There exists a blocking chain, either on $P[u, v]$ or $P[v, u]$, for any pair of invisible vertices (u, v) of P.*

Proof: Let x be a point of the open set (u, v) such that $x \notin \Gamma P$. Such a x must clearly exist because $\overline{uv} \not\subset \Gamma P$. Let $\mathcal{T}_c$ denote the number of subchains of chain c of type $\mathcal{T}$ which enclose x, where $c \in \{uv, vu\}$ and $\mathcal{T} \in \{\mathcal{RF}, \mathcal{RB}, \mathcal{LF}, \mathcal{LB}\}$. Here we use '$uv$' and '$vu$' for convenience to denote $P[u, v]$ and $P[v, u]$ respectively. Suppose $\mathcal{RF}_{uv} > \mathcal{RB}_{uv}$. Elementary combinatorics will now tell us that in any sequence (induced by $\prec$) of the $\mathcal{R}$-chains of $P[u, v]$ enclosing x, there is at least one occurrence of an $\mathcal{RF}$-chain which is not followed immediately by an $\mathcal{RB}$-chain. Such a chain is clearly a blocking chain of (u, v).

So let $\mathcal{RF}_{uv} \leq \mathcal{RB}_{uv}$. But since u and v lie on either side of x on $\overleftrightarrow{uv}$, clearly
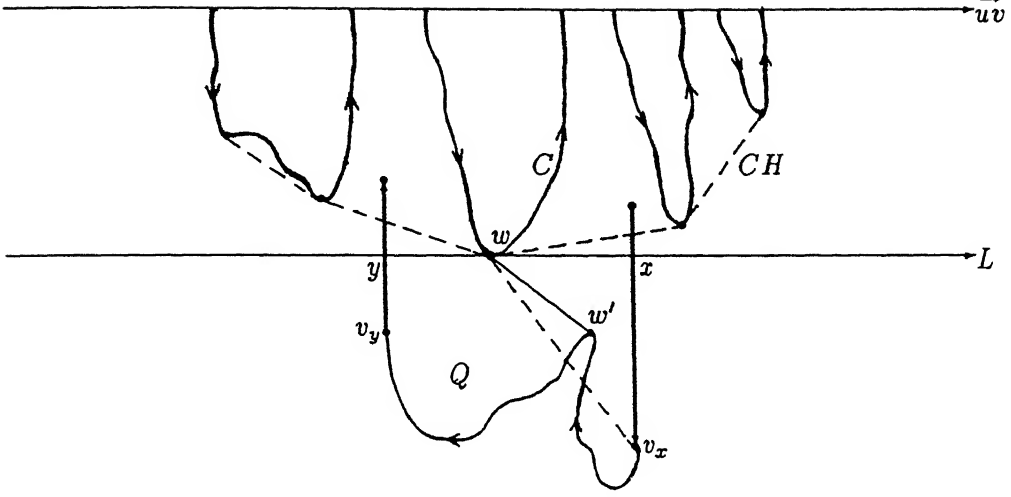
Figure 2.2: The vertex w of CH farthest from $\vec{uv}$ is a blocking vertex of (u, v).

the number of $\mathcal{F}$-chains of $P[u, v]$ enclosing x is exactly one more than the number of $\mathcal{B}$-chains. Thus $\mathcal{RF}_{uv} + \mathcal{LF}_{uv} = \mathcal{RB}_{uv} + \mathcal{LB}_{uv} + 1$. This implies that $\mathcal{LF}_{uv} > \mathcal{LB}_{uv}$. Now applying Lemma 2.2.1 to the left side of $\vec{uv}$ we find $\mathcal{LF}_{uv} + \mathcal{RB}_{vu} = \mathcal{LB}_{uv} + \mathcal{RF}_{vu}$. Hence $\mathcal{RF}_{vu} > \mathcal{RB}_{vu}$. From our earlier observation for the case $\mathcal{RF}_{uv} > \mathcal{RB}_{uv}$, we see that $P[v, u]$ must now have a blocking chain of (u, v). ∎

Lemma 2.2.3: *If $P[u, v]$ contains blocking chains of an invisible pair of vertices (u, v), then there exists a blocking vertex of (u, v) on the convex hull of all the blocking chains of (u, v) in $P[u, v]$.*

Proof: Let w be the vertex on the convex hull CH of the blocking chains which admits a tangent to the hull parallel to $\vec{uv}$. We show that w is a blocking vertex of (u, v). Let the tangent through w parallel to $\vec{uv}$ be L and let the $\mathcal{RF}$-chain containing w be C. See Figure 2.2. We first show that there exists a vertex $w' \in P(v, u)$ on the right of or on L such that $\overline{ww'} \subset \Gamma P$. Let x and y be the points in $L \cap P$, closest to w on either side of it. Clearly $\overline{xy} \subset \Gamma P$ because P is oriented clockwise and w lies on an $\mathcal{RF}$-chain. Suppose $x \in P(u, v)$ and let C_x be the chain of $P[u, v]$ containing x. Since $x \notin \Gamma CH$ clearly $C_x \not\subset \mathrm{span}(C)$. If
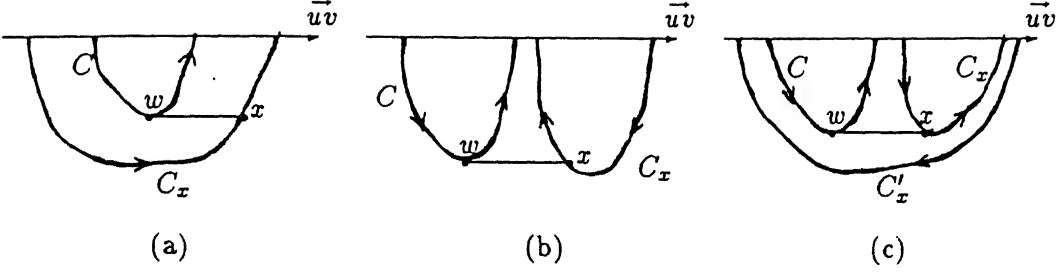
Figure 2.3: The points x and y have to be in $P[v, u]$

however $C \prec C_x$, then because $\overline{wx} \subset \Gamma P$, C_x must be the immediate superior of C. See Figure 2.3a. C being a blocking chain it now follows that C_x must be an $\mathcal{RF}$-chain. But this cannot be because now for $\overline{wx}$ to be in ΓP, ΨP must lie on the left of C_x contrary to our assumption that P is oriented clockwise. The only other possibility is where span$(C_x) \cap$ span$(C) = \phi$. Suppose C_x is an $\mathcal{RB}$-chain as in Figure 2.3b. Here too from the clockwise orientation of P it follows that this is not possible. Finally let C_x be an $\mathcal{RF}$-chain. See Figure 2.3c. Since $x \notin \Gamma CH$, C_x is clearly not a blocking chain implying that it does have an immediate $\mathcal{RB}$-superior C_x'. Now if $\overline{wx} \subset \Gamma P$, it is easy to see that no $\mathcal{R}$-chains of $P[u, v]$ can be such that it encloses just one of the two, C_x and C, and not the other. Thus C_x' is an immediate superior of C as well contradicting the fact that C is a blocking chain. Thus $x \in P(v, u)$. Similarly we can show that $y \in P(v, u)$.

If either x or y is a vertex of P, we take w' as that vertex. Otherwise let v_x and v_y be respectively the endpoints of the edges of P containing x and y such that both v_x and v_y lie on the right of L (see Figure 2.2). We now observe that since $\overline{wx} \subset \Gamma P$, P must be directed from x to v_x at x and from v_y to y at y. This along with the fact that both x and y are in $P[v, u]$, it is clear that the chain $P[x, y]$ which contains v_x and v_y is a subset of $P[v, u]$. Therefore $Q = P[x, y] \cup \overline{xy}$ is a simple polygon with $\Gamma Q \subset \Gamma P$. We will now choose a vertex of $P(x, y)$ as our w'.

If $\overline{wv_x} \subset \Gamma Q$ then $w' = v_x$. Otherwise there must exist vertices of Q in the interior of the triangle $\triangle wv_x x$. In that case we take w' as the vertex z of Q in $\triangle wv_x x$ for which

the angle between 'positive' L and $\vec{wz}$ is the least. That such a w' is just what we want, can be verified easily.

Now to show that w is a blocking vertex we first observe that since P is a simple polygon with $\overline{ww'}$ its chord, for any pair of vertices (u', v') with $u' \in P(w, w')$ and $v' \in P(w', w)$, if $\overline{u'v'} \subset \Gamma P$ then it must be that $(u', v') \cap (w, w') \neq \phi$. We now show that for any pair of vertices (u', v') with $u', v' \in P[u, v] - w$ and $\overline{u'v'} \subset \Gamma P$, $(u', v') \cap (w, w') = \phi$ thus showing that w is indeed a blocking vertex of (u, v).

Firstly for a pair of vertices (u', v'), both of which are either to the left of or on L, obviously $(u', v') \cap (w, w') = \phi$. So without loss of generality let u' be on the right of but not on L. Let C' be the $\mathcal{R}$-chain of $P[u, v]$ containing u' and since $u' \notin \Gamma CH$, clearly $C' \not\subset \mathrm{span}(C)$. Here again we have four cases similar to those in Figure 2.3.

1. $C \prec C'$

   (a) C' is an $\mathcal{RF}$-chain (Figure 2.4a):

   $\overline{ww'} \subset \mathrm{span}(C')$ and $\overline{u'v'} \cap \mathrm{span}(C') = \phi$ because $\overline{ww'}, \overline{u'v'} \subset \Gamma P$ and P is oriented clockwise.

   (b) C' is an $\mathcal{RB}$-chain (Figure 2.4b):

   C being a blocking chain, there is an immediate $\mathcal{RF}$-superior C_1 of C in $P[u, v]$ such that $C \prec C_1 \prec C'$. Again because $\overline{ww'}, \overline{u'v'} \subset \Gamma P$ we find $\overline{ww'} \subset \mathrm{span}(C_1)$ and $\overline{u'v'} \cap \mathrm{span}(C_1) = \phi$.

2. $\mathrm{span}(C) \cap \mathrm{span}(C') = \phi$

   (a) C' is an $\mathcal{RF}$-chain (Figure 2.4c):

   We know that C is a blocking chain and that C' is not. From these it is easy to infer that there must be a chain C_1 enclosing just one of these two and not the other. Therefore, assume $C \prec C_1$ without loss of generality. Again because $\overline{ww'}, \overline{u'v'} \subset \Gamma P$ we find $\overline{ww'} \subset \mathrm{span}(C_1)$ and $\overline{u'v'} \cap \mathrm{span}(C_1) = \phi$.

(a)

(b)

(c)

(d)
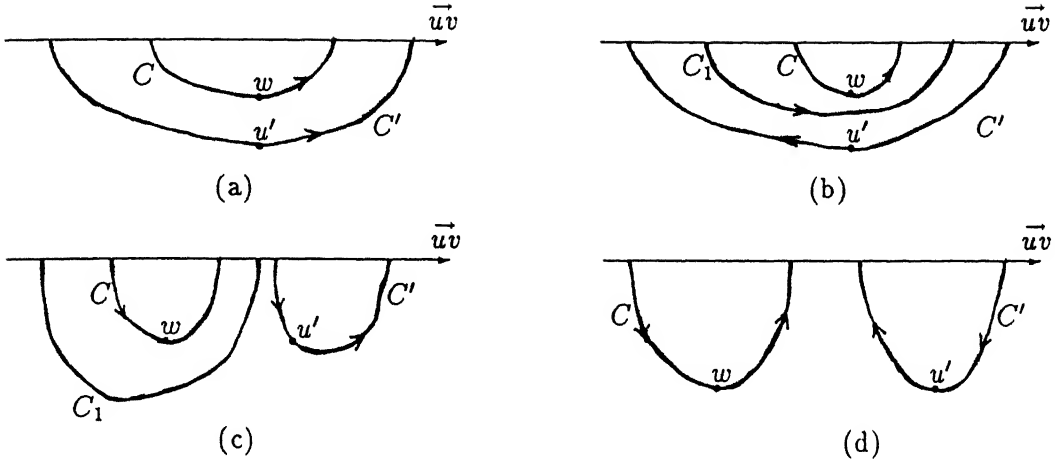
Figure 2.4: For every pair (u', v'), $u'v' \in P[u, v] - w$, if $\overline{u'v'} \subset \Gamma P$ then $\overline{ww'} \cap \overline{u'v'} = \phi$

(b) C' is an $\mathcal{RB}$-chain (Figure 2.4d):

Since P is clockwise oriented, clearly either v' is on the left of $\vec{uv}$ or $[u', v'] - u' \subset \text{span}(C')$. Moreover $\overline{ww'} \cap (\text{span}(C') \cup C') = \phi$.

It is trivially seen that in each of the above cases $\overline{u'v'} \cap \overline{ww'} = \phi$. The lemma is thus proven. ∎

Now our assignment of blocking vertices to invisible pairs of P is just that implied by Lemma 2.2.3, i.e., for an invisible pair (u, v) we pick any one of the two chains $P[u, v]$ and $P[v, u]$ which contains blocking chains of (u, v), compute the convex hull of all the blocking chains on it, and then assign the vertex on the hull which admits a tangent to the hull parallel to $\vec{uv}$ as the blocking vertex of (u, v). All the vertices on the convex hull of the blocking chains are clearly reflex vertices of P. Thus our assignment assigns only reflex vertices of P as blocking vertices of invisible pairs. In fact we show in the following lemma that even the converse of this is true. Henceforth we assume that B is our assignment of blocking vertices to invisible pairs as described above.

Lemma 2.2.4: *For any three consecutive vertices u, v and w of P such that $w \in P(u, v)$,*
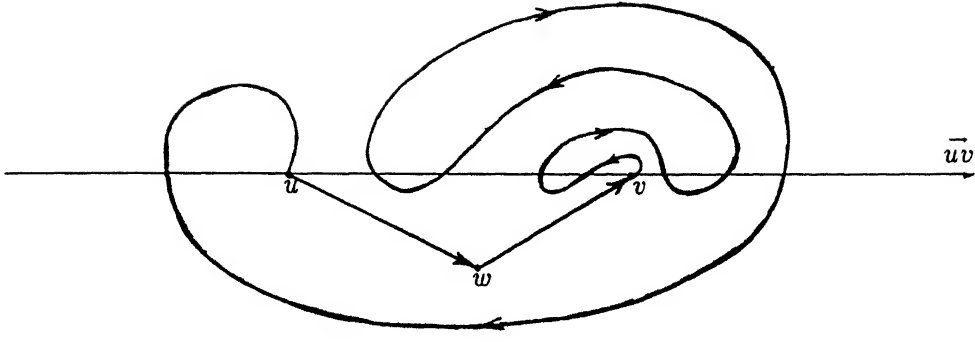
Figure 2.5: Every concave vertex of P is assigned as the blocking vertex of some invisible pair.

if w is a reflex vertex of P then $B((u,v)) = w$. Note that if w is a reflex vertex then (u,v) is necessarily an invisible pair.

Proof: Consider any $\mathcal{RF}$-chain in $P[v,u]$. We know from Lemma 2.2.1 that any such chain must certainly have an immediate superior chain in either $P[u,v]$ or $P[v,u]$ which is an $\mathcal{RB}$-chain with respect to $\overleftarrow{uv}$ (see Figure 2.5). This can obviously not be in $P[u,v]$ as $P[u,v]$ has exactly one vertex w which is on the left of $\overleftarrow{uv}$. Thus every $\mathcal{RF}$-chain in $P[v,u]$ has an immediate $\mathcal{RB}$-superior in $P[v,u]$. Hence $P[v,u]$ has no blocking chain of (u,v). It is now clear that $B((u,v)) = w$. ∎

We can further show that the assignment B is in fact consistent.

Lemma 2.2.5: *The assignment B of blocking vertices to invisible pairs is consistent.*

Proof: Let w be the blocking vertex assigned to an invisible pair (u,v) and let (u',v') be another invisible pair separable from (u,v) with respect to w. Let us assume without loss of generality that $w \in P(u,v)$ and that $P[u',v'] \subset P[u,w]$. We now show that $B((u',v')) \neq w$ hence proving the consistency.

Suppose on the contrary that $B((u',v')) = w$. Then w lies on blocking chains of both $P[u,v]$ and $P[v',u']$. Let these two $\mathcal{RF}$-chains with respect to $\overrightarrow{uv}$ and $\overleftarrow{u'v'}$ be respectively $C_1 = P[x_1, y_1]$ and $C_2 = P[x_2, y_2]$. Clearly $C_1 \cap C_2 \neq \phi$ because w lies on both. Also $C_1 \cap C_2$ is a contiguous chain of P.

25

Figure 2.6: If $C \subset P[u, x_2]$ then $p \notin (y_1^+)$.

We prove the lemma through a case analysis. We first assume that $C_1 \not\subset C_2$. If this is not true we could carry on with the same proof simply by interchanging the roles of C_1 and C_2. So we essentially have two cases here:

1. C_2 contains neither x_1 nor y_1.

2. C_2 contains exactly one of x_1 and y_1.

We treat these two cases separately below.

case 1: C_2 contains neither x_1 nor y_1.

Note that then $C_2 \subset C_1$. Lemma 2.2.1 assures us that there exists an immediate superior $C = P[x, y]$ of C_2 in P with respect to $\overrightarrow{v'u'}$ and that is necessarily a B-chain. Suppose $C \subset P[u, x_2]$. Let $p = \text{last}(P[u, x] \cap \overrightarrow{uv})$.

case 1.1: $p \in (y_1^+)$.

In this case $P(y, x_2) \cap \overrightarrow{uv} \neq \phi$ because otherwise C_1 would be $P[p, y_1]$ which is a B-chain. Let $q = \text{first}(P(y, x_2) \cap \overrightarrow{uv})$. If $q \in (x_1^-)$ then $P[p, q]$ is an immediate $\mathcal{RB}$-superior of C_1 in $P[u, v]$ and if $q \in (y_1^+)$ then actually $q \in (p^+)$ making $P[p, q]$ an $\mathcal{RF}$-chain of $P[u, v]$. See Figure 2.6 in which these two possible positions of q are shown as q_1 and q_2 respectively. Both the above statements follow from the facts that $C \subset P[p, q]$ and that C is the immediate superior of C_2 with respect to $\overrightarrow{v'u'}$. Now the former contradicts our

26

Figure 2.7: If $C \subset P[u, x_2]$ then $p \notin [x_1^-]$.

assumption that C_1 is a blocking chain of (u, v). As for the latter, again because C is an immediate superior of C_2, any immediate superior of $P[p, q]$ with respect to $\vec{uv}$ is also an immediate superior of C_1. Thus when C_1 is a blocking chain, so is $P[p, q]$. If that is so, then $B((u, v)) = w$ only when w lies on the convex hull of $C_1 \cup P[p, q]$. This cannot be true since w lies in the interior of the polygon $P[p, y] \cup \overline{yx'} \cup P[x_1, x'] \cup \overline{x_1 p}$ which in turn is contained in the convex hull where $x' = \text{first}((y^+) \cap C_1)$.

case 1.2: $p \in [x_1^-]$.

Here if $P[p, y] \subset P[x_1, x_2]$ then obviously $x_1 = p$. Now again from an argument identical to that above we find that w cannot then be on the convex hull of C_1 (remember that now $C \subset C_1$) preventing it from being assigned as the blocking vertex of (u, v). See Figure 2.7a. Hence $P[p, y] \subset P[u, x_1]$ and since P is in general position it must be that $P(y, x_1) \cap \vec{uv} \neq \phi$. Thus let $q = \text{first}(P(y, x_1) \cap \vec{uv})$. Again because C is the immediate superior of C_2 with respect to $\vec{v'u'}$, it is easy to see that $q \in (p_1, x_1)$. See Figure 2.7b. So $P[p, q]$ is an $\mathcal{RF}$-chain $P[u, v]$. Just as we argued earlier, clearly here too $P[p, q]$ is a blocking chain and w cannot be on the convex hull of $P[p, q] \cup C_1$ thus contradicting our choice of w.

27

We thus find that our initial assumption that $C \subset P[u, x_2]$ cannot hold. So $C \subset P[x_2, u] \subset P[v', u']$. Therefore C is an $\mathcal{RB}$-chain of $P[v', u']$ but it is also the immediate superior of C_2. Then C_2 is not a blocking chain—a contradiction.

case 2: C_2 contains only one of x_1 and y_1.

We prove the claim for $x_1 \in C_2$. The case $y_1 \in C_2$ is similar.

Consider a wedge formed by the intersection of half-planes defined by a pair of rays L_1 and L_2 such that $L_1 \cap L_2 = O$. Just as we observed for the $\mathcal{R}$- and $\mathcal{L}$-chains of P with respect to a directed line, here too there clearly exists a natural total ordering of the chains of P with one end point each on L_1 and L_2 and wholly contained in the interior of the wedge (henceforth we call such chains as those which *cross* the wedge). We order such chains with respect to the wedge starting from the apex O. An *immediate superior* of a chain and the *highest* and *lowest* among chains in this context have the obvious interpretations. It is also easy to see that there exists an analogue of Lemma 2.2.1 for wedges also, taking the apex of the wedge as the point x of the lemma. In fact the proof of Lemma 2.2.1 would go through as it is for wedges as well. Henceforth all our references to Lemma 2.2.1 would mean Lemma 2.2.1 applied to both lines and wedges.

Applying Lemma 2.2.1 to the wedge $\mathcal{W}$ formed by the right half-planes of $\overrightarrow{uv}$ and $\overrightarrow{v'u'}$ we see that there must exist a chain $C = P[x, y]$ with $x \in (y_2^+)$ and $y \in (x_1^-)$ (see Figure 2.8) which is the immediate superior of C_2 with respect to $\mathcal{W}$.

An argument almost identical to that in case 1.1 will imply that $C \subset P[v', u']$. Now just as in the earlier cases, here too we find that if $q = \text{first}(P[y, v'] \cap \overrightarrow{v'u'}) \in (x_2^-)$ (shown in Figure 2.8 as q_1), then C_2 is not a blocking chain ($P[x, q]$ is its immediate $\mathcal{RB}$-superior) and if $q \in (y_2^+)$ (shown as q_2) then $q \in (x^+)$ implying that $P[x, q]$ is also a blocking chain with w no more on the convex hull of $C_2 \cup P[x, q]$, in either case contradicting our earlier assumptions.

Hence the lemma. ∎
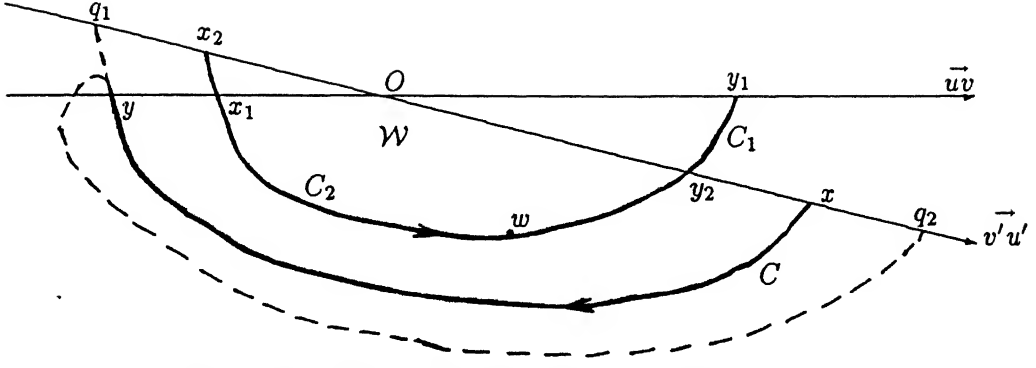
Figure 2.8: $B((u', v')) \neq w$ when C_2 contains x_1 but not y_1.

We have thus proved Theorem 2.1 by constructing a consistent invisible pair to blocking vertex assignment for any simple polygon P. Similar to Everett's restatement of Ghosh's Necessary Condition S4 [19, section 2.2.4], we can now state a version of Theorem 2.1 which throws light on the kind of vertices one can expect those corresponding to the blocking vertices assigned to the invisible pairs of a graph to be, in a polygon whose visibility graph is the given one. Our restatement follows easily from Lemma 2.2.4.

Theorem 2.2: *Let G be the visibility graph of a simple polygon and let P be any polygon in general position with G as its visibility graph. Then there exists a consistent assignment B of blocking vertices to invisible pairs of G such that $B((u, v)) = w$ for some invisible pair (u, v) if and only if w corresponds to a reflex vertex of P.*

29

Orthogonal Edge Visibility Graphs — an Introduction

In the next two chapters we study a class of visibility graphs which are intended to capture the visibility relationships among pairs of edges of an orthogonal polygon with or without holes. In these chapters we extend some of the results on orthogonal edge visibility graphs that had been obtained earlier by O'Rourke [36, section 7.3]. For convenience, we use the terms 'polygon' and 'visibility graph' throughout this introduction and the next two chapters to mean 'orthogonal polygon with or without holes' and 'orthogonal edge visibility graph' respectively.

The visibility graph of a given polygon P consists of a vertex e for each edge e^* of P with two vertices e and f of G being adjacent if the two corresponding edges e^* and f^* can 'see' (we will define this in a short while) each other. The visibility graph consists of two disjoint components—the *vertical* and the *horizontal*. The vertical(horizontal) edge visibility graph of P is the graph G with a vertex e for every horizontal(vertical) edge e^* of P such that e and f are adjacent in G if and only if there exists a rectangle of non-zero area, wholly contained in ΓP with two opposite sides of the rectangle on e^* and f^* (e^* and f^* are then said to *see* or *be visible to* each other). That the two components—horizontal and vertical—of the visibility graph are disjoint follows trivially from our definition. We sometimes say that e^* and f^* can see each other along a line $x = x_0$ if the open vertical segment

at $x = x_0$ joining e^* and f^* lies entirely in ΨP. Note that the word 'vertical(horizontal)' in the term 'vertical(horizontal) visibility graph' refers to the direction of visibility and not to the direction of the edges whose mutual visibility is being talked about. O'Rourke [36, Lemma 7.3] has shown that each of the components viz., horizontal and the vertical, of the visibility graph of a polygon without holes, is connected. The proof however applies just as well to polygons with holes. The visibility graph therefore consists of two connected components. We say that a polygon *realizes* a graph G if either

- G is connected and is either the horizontal or the vertical visibility graph of P, or

- G is the visibility graph of P.

A graph G is said to be realizable if there exists a polygon which realizes G.

We devote one each of the next two chapters to a study of the above two notions of realizability. In chapter 3 we study the realizability of a given connected graph as a polygon, either the horizontal or the vertical visibility graph of which is the given one. As for the second notion, following O'Rourke [36], we prefer to look at it as the realizability of a *pair* of graphs i.e., two disjoint connected graphs G_1 and G_2 are said to be *meshable* or *jointly realizable* if there exists a polygon which realizes $G_1 \cup G_2$. Note that in an orthogonal polygon the number of horizontal edges is equal to the number of vertical edges. Hence two graphs can mesh only if they have the same number of vertices. We study this problem in chapter 4 where we try to characterize pairs of meshable graphs (we assume by default that the two graphs have an equal number of vertices). In fact, we investigate only the visibility graphs of polygons without holes, and even for this apparently simpler case, a complete characterization of the meshability conditions seems to be a rather formidable task. O'Rourke [36, Lemma 7.3] has shown that the visibility graph components for polygons without holes are trees. Thus in chapter 4 we study meshability among pairs of trees.

31

Chapter 3

Orthogonal Edge Visibility Graphs — I

In this chapter, we present some results on determining whether a given connected graph is realizable under the first of the realizability notions we referred to in the introduction to Orthogonal Edge Visibility Graphs. O'Rourke [36] had earlier obtained results on this problem for polygons without holes. In fact the results of O'Rourke can be summarized as below:

([36, Lemmas 7.3 and 7.4]) A connected graph is realizable if and only if it is a tree.

We extend the above to polygons with holes in this chapter. In the rest of this chapter, the term 'polygon' should be taken to mean an 'orthogonal polygon with holes' and the term 'realizable' interpreted accordingly.

In section 1 of this chapter we introduce some definitions and notations that we will need. In section 2 we derive our set of six necessary conditions for a connected graph to be realizable. In section 3 we show that this set of conditions is not sufficient but is sufficient *upto leaves.*

3.1 Definitions and Notations

In what follows we study without loss of generality only the vertical visibility graphs. Whatever we prove for these obviously holds for the horizontal visibility graphs too. Henceforth, we will use the terms *edge* and *visibility graph* to mean a *horizontal* edge and *vertical visibility graph* respectively. An edge of the polygon P is said to be facing upward(downward) if ΨP is above(below) the edge. The closed interval on the X-axis covered by a segment g is denoted $[I_g]$ (or (I_g) for the open interval) and if g is horizontal then y_g is used to denote the ordinate of g. The corresponding notations for an edge e^* of P are $[I_e]$ $((I_e))$ and y_e respectively. Also for an X-interval I let $l(I)$ and $r(I)$ denote respectively the abcissae of the left and the right end points of I. Finally, to make our notations less clumsy we abbreviate $[I_1] \otimes [I_2]$ to $[I_1 \otimes I_2]$ where $\otimes$ denotes the usual set operations, for any two X-intervals $[I_1]$ and $[I_2]$. We also use a similar abbreviation for open intervals.

3.2 Necessary Conditions

In this section, we derive a set of necessary conditions for a graph to be realizable. Let G be the visibility graph of a polygon P in general position. Any property proven on G without making any assumptions about P will obviously constitute a necessary condition. We establish an independent set of six such properties (the word 'independent' is used to mean that none of the properties follows from the other five) of G in this section. For this we first make a few rather easy but useful observations which we exploit repeatedly in the proofs of the properties of G that follow.

Observation 3.2.1: *Given an edge e^* of P and $x_0 \in (I_e)$, the edge of P which can see e^* along $x = x_0$ is unique.*

The next two observations concern a pair of edges e^* and f^* of P such that e^* faces upward(downward) and f^* is above(below) e^*.
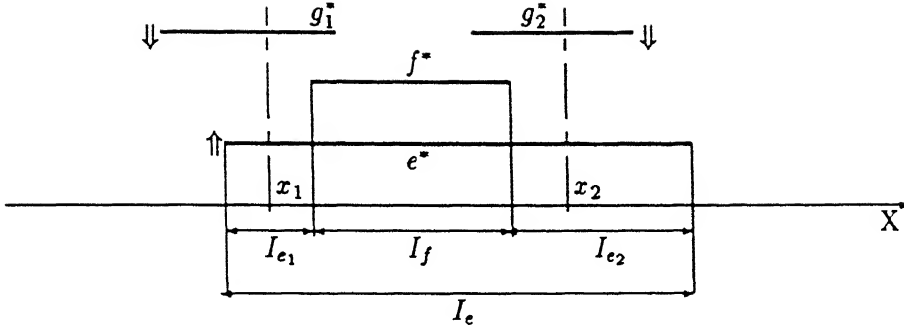
Figure 3.1: Existence of a leaf adjacent to e when $[I_f] \subset (I_e)$

Observation 3.2.2: *If $(I_e \cap I_f) \neq \phi$ then for any $x_0 \in (I_e \cap I_f)$, the edge g^* which can see e^* along $x = x_0$ is such that $y_f \geq y_g > y_e$ ($y_e > y_g \geq y_f$) where the equality holds only when $g \equiv f$.*

Observation 3.2.3: *If $[I_e - I_f]$ has two connected components $[I_{e_1})$ and $(I_{e_2}]$ (in this case $[I_f] \subset (I_e)$) and there exist $x_1 \in (I_{e_1})$ and $x_2 \in (I_{e_2})$ (see Figure 3.1) such that for the edges g_1^* and g_2^* which can see e^* along $x = x_1$ and $x = x_2$ respectively, $y_{g_1}, y_{g_2} > y_f$ ($y_{g_1}, y_{g_2} < y_f$), then the closest edge g^* of P to e^*, in the region bounded by $x = x_1$, $x = x_2$ and e^*, sees e^* but no other edge of P. The vertex g of G is thus a leaf adjacent to e.*

We are now ready to prove the properties of G. For completeness, we list the property that G must connected—one which we have already mentioned—also as a necessary condition here.

Necessary Condition O1: *G must be connected.*

Necessary Condition O2: *G must be bipartite.*

Proof: This is obvious from the fact that no two edges, both facing upward or both facing downward can see each other. The partitions of G thus correspond to the sets of edges of P facing upward and those facing downward. ∎

34

Necessary Condition O3: *G must be planar.*

Proof: We prove this by exhibiting a planar embedding of G i.e., a map $M(G) \subset \Pi$ mapping the vertices of G to points in the plane Π and the edges to simple curves whose end points are the images of the end-vertices of the edge, such that

- M restricted to $V(G)$ is one-to-one and

- for any two distinct edges $(e, f), (p, q) \in E(G)$, $(M(e, f) - \{e, f\}) \cap M(p, q) = \phi$ and $(M(p, q) - \{p, q\}) \cap M(e, f) = \phi$, where $M(e, f)$ denotes the image of the edge (e, f) of G under M.

First for every pair of visible edges e^* and f^* of P (or equivalently, adjacent vertices e and f of G) we choose a 'line-of-sight' (denoted henceforth as LOS_{ef} for the pair (e, f)) which is a vertical line at $x = x_{ef}$ from e^* to f^* such that there exists a rectangle bounded by $x = l_{ef}$, f^*, $x = r_{ef}$ and e^* and wholly contained in ΓP for some $l_{ef} < x_{ef} < r_{ef}$ and $l_{ef}, r_{ef} \in (I_e \cap I_f)$. This is obviously possible for any pair of visible edges.

Let e^* be an edge facing upward and let a^* be the edge visible to e^* and closest to it. We now mark as many distinct points in the interior of the segment joining the points with coordinates (l_{ea}, y_e) and (l_{ea}, y_a) as the number of edges of P visible to e^* whose chosen lines-of-sight to e^* are to the left of x_{ef}. We also similarly mark points in the segment joining the points (r_{ea}, y_e) and (r_{ea}, y_a). See Figure 3.2. This is done for each edge of P—note that this can be done *independently* for each of the edges. We are now ready to describe the embedding M which maps G into the plane.

For the vertex e of G, $M(e)$ is the point with coordinates (x_{ea}, y_e) viz., the point $LOS_{ea} \cap e^*$. For an edge (e, f) between the vertices e and f, $M(e, f)$ coincides with LOS_{ef} except possibly in the vicinities of e^* and f^*. We describe the nature of these 'deviations' from LOS_{ef} in the vicinity of e^*. That near f^* is similar. If $f = a$ then there is no deviation i.e., $M(e, f)$ is LOS_{ef} itself in the vicinity of e^* too. Otherwise let LOS_{ef} be the kth line-of-sight of e^* starting from $M(e)$ to its right. In the vicinity of e^* then $M(e, f)$ consists of

35
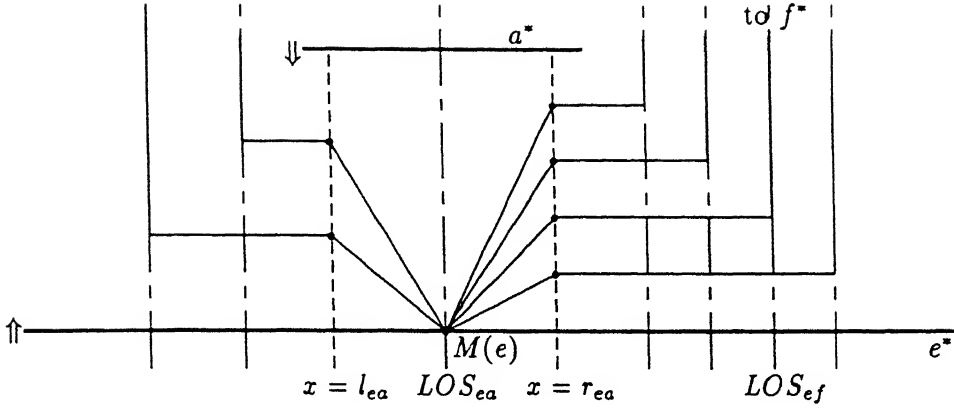
Figure 3.2: The embedding $M(G)$ in the vicinity of e^*

a straight segment joining $M(e)$ to the kth marked point from the top (bottom if e^* faces downward) on $x = r_{ea}$ and a horizontal segment from there to the right upto LOS_{ef}. Thus at worst $M(e, f)$ consists of five pieces $s_{ef}, h_{ef}, v_{ef}, h_{fe}, s_{fe}$ in that order from e^* to f^* where s_{ef} and s_{fe} are the 'slant' segments of $M(e, f)$ at the e^* and f^* ends respectively, h_{ef} and h_{fe} are the horizontal segments and v_{ef} the vertical segment coincident with LOS_{ef}. It is clear that for any visible pair (e, f), $(M(e, f) - \{e, f\}) \subset \Psi P$.

We will now show that $M(G)$ is a planar embedding, viz., for any two distinct edges (e, f) and (p, q) of G, $M(e, f)$ and $M(p, q)$ do not intersect anywhere except possibly at a common endpoint if there is one. In the rest of the proof the word 'intersection' is used to mean 'intersection of $M(p, q)$ with $M(e, f)$ at a point other than a common end point'.

For any edge e^*, the closest visible edge to which is a^*, let R_e denote the rectangle bounded by e^*, $x = l(I_e \cap I_a)$, a^* and $x = r(I_e \cap I_a)$. Obviously, from Observation 3.2.2, R_e is $empty$ (we use this term to mean that the interior of the rectangle lies wholly in ΨP) for every edge e^*. Also clearly $\Gamma R_e \cap \Gamma R_p = \phi$ for any two distinct edges e^* and p^* unless they are the closest visible edges to each other. Moreover it is also easy to see that for $\{p, q\} \neq \{a, e\}$, $v_{pq} \cap \Psi R_e = \phi$. We prove the rest through the following two claims.

claim 1: $\{(s_{ef} \cup h_{ef}) \cap (s_{pq} \cup h_{pq})\} - \{e\} = \phi.$

36
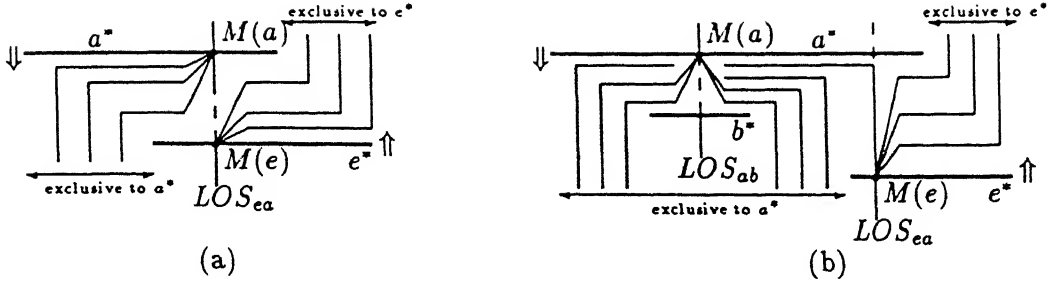
Figure 3.3: The two sides of LOS_{ea} are exclusive to either a^* or e^* ($l(I_a) < l(I_e)$ and $r(I_a) \in (I_e)$).

The claim is obviously true, from our construction, for $p = e$. So let $p \neq e$.

Suppose $p = a$. If $(I_a) \subset (I_e)$ then Observation 3.2.3 tells us that a is a leaf of G (a^* sees only e^* in P). In that case clearly $s_{aq} \cup h_{aq} = \phi$. Similarly if $(I_e) \subset (I_a)$ then $s_{ef} \cup h_{ef} = \phi$. In either case our claim is trivially true. So assume that neither of (I_e) and (I_a) is a subset of the other implying that either $l(I_a) < l(I_e)$ and $r(I_a) \in (I_e)$ or $r(I_a) > r(I_e)$ and $l(I_a) \in (I_e)$. Now in the former case no LOS_{aq} can lie on the right of LOS_{ea} and no LOS_{eq} can lie on its left. Similarly in the latter case no LOS_{aq} can lie on the left of and no LOS_{eq} on the right of LOS_{ea}. All these easily follow from the fact that a^* is the closest visible edge to e^*. Figure 3.3 (a) when e^* is also the closest to a^* and (b) when some $b^* \neq e^*$ is the closest to a^*, illustrates this for the first case. Thus the two sides of LOS_{ea} are in some sense 'exclusive' to the lines-of-sight of either a^* or e^*, without 'intrusions' from the other. The claim now follows easily.

If however $p \neq a$ then $\Gamma R_e \cap \Gamma R_p = \phi$. Thus since $s_{pq} \subset \Gamma R_p$ and $s_{ef} \subset \Gamma R_e$, obviously $s_{pq} \cap s_{ef} = \phi$. Suppose $s_{ef} \cap h_{pq} \neq \phi$. Then obviously $h_{pq} \cap \Psi R_e \neq \phi$. So $y_e < y_{h_{pq}} < y_a$, $(I_{h_{pq}} \cap I_e) \neq \phi$ and $(I_{h_{pq}} \cap I_a) \neq \phi$. From our construction it is clear that $[I_{h_{pq}}] \subset (I_p)$. Moreover since a^* is the closest visible edge to e^*, from Observation 3.2.2 either $y_p > y_a$ or $y_p < y_e$. In the former case we would have $y_b > y_a$ and in the latter $y_b < y_e$ where b^* is the closest visible edge of P to p^*. Again our construction of $M(G)$ implies easily that $y_{h_{pq}}$ is between y_p and y_b. Thus either $y_{h_{pq}} > y_a$ or $y_{h_{pq}} < y_e$. In either

case it is a contradiction. The claim is thus settled.

To complete the proof of planarity of $M(G)$ we only need to show that no intersection can occur between h_{pq} and v_{ef} or vice-versa. Note that no two horizontal and no two vertical segments can intersect each other. Also neither can two horizontal segments coincide with each other over some interval if for every such pair of edges of G, $h_{pq} \cap v_{ef} = \phi$.

Suppose v_{ef} intersects h_{pq}. Then LOS_{ef} too intersects h_{pq}. We already know that $[I_{h_{pq}}] \subset (I_p)$. So $x_{ef} \in (I_p)$. Hence clearly either y_e or y_f lies between $y_{h_{pq}}$ and y_p and if $y_e(y_f)$ is such then $(I_e \cap I_p) \neq \phi$ $((I_f \cap I_p) \neq \phi)$. From Observation 3.2.2 again it now follows that the closest visible edge to p^* must lie between y_p and $y_{h_{pq}}$, which is clearly absurd.

Thus $M(G)$ is indeed a planar embedding making G a planar graph. ∎

Necessary Condition O4: *If G is a tree then there must exist a pair of leaves of $G separated by a distance of three on G.*

Proof: Let v^* be the leftmost vertical hole edge of P. Imagine sweeping a vertical line L to the right starting from v^*. We first build two sequences S_1 and S_2 of vertices of G as L sweeps across P. For this let the edges adjacent to v^* be h_u^* and h_d^* with h_u^* above h_d^*. Also let b_u^* and b_d^* be the edges cutting L which are closest to v^* above and below it respectively, at the start of the sweep. See Figure 3.4. Note that because v^* is the leftmost hole edge, b_u^* and b_d^* belong to the outer boundary of P. The rest of the sweep is described formally in Figure 3.6. We use b_u (b_d, h_u and h_d respectively) to denote both the point $L \cap b_u^*$ ($L \cap b_d^*$, $L \cap h_u^*$ and $L \cap h_d^*$ respectively) and the vertex of G corresponding to b_u^* (b_d^*, h_u^* and h_d^* respectively).

From the remarks made in the description of Procedure Sweep it is easy to see that the sweep does terminate and that at the end of it S_1 and S_2 contain simple paths of G. Moreover from our start and terminating conditions it is also clear that first(S_1) can see first(S_2) and last(S_1) can see last(S_2) where first(S_i) and last(S_i) refer to the first and the last elements of S_i, $i = 1, 2$. Thus S_1 along with the list S_2 reversed forms a cycle of G
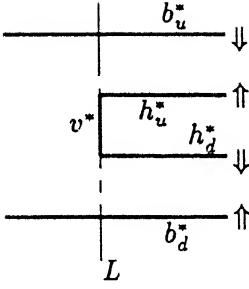
Figure 3.4: The start of Algorithm Sweep.

Figure 3.5: Sweep update when v^* is adjacent to (a) b_u^* (b) h_u^*, in P
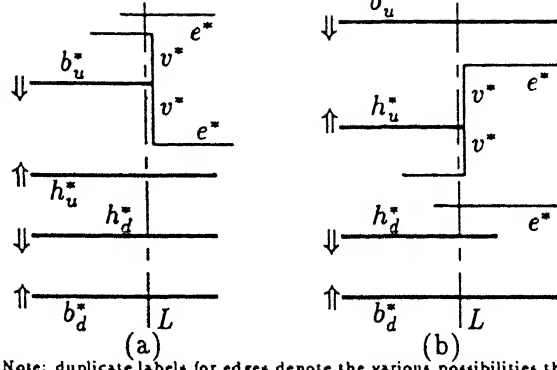
Note: duplicate labels for edges denote the various possibilities that can occur

unless first(S_1) = last(S_1) and first(S_2) = last(S_2) (note that then we would have a 'cycle' of length two). But we have assumed that G is a tree and hence cycle free. So let $p =$ first(S_1) = last(S_1) and $q =$ first(S_2) = last(S_2) with possible lines-of-sight between p^* and q^* at 'either end'. Now from Observation 3.2.3 it is easy to infer that p and q do have a leaf adjacent to each in G. Note that there is at least one hole of P between p^* and q^*. G must therefore have a pair of leaves separated by a distance of three on G. ∎

Necessary Condition O5: *Every four-cycle of G must have a leaf adjacent to one of the vertices of the cycle.*

Proof: Consider the four-cycle shown in Figure 3.8. Assume without loss of generality that e^* faces upward. Thus g^* also faces upward and h^* and f^* face downward. Then clearly $\min(y_h, y_f) > \max(y_e, y_g)$. Also say $y_e < y_g$ and $y_h < y_f$. Note that this means $[I_e] \not\subseteq [I_g]$ and $[I_f] \not\subseteq [I_h]$ because otherwise $e^*(f^*)$ will not be able to see either of h^* and $f^*(e^*$ and $g^*)$. We now have two possibilities.

case 1: $[I_g] \subset (I_e)$

Then to see both e^* and g^* both (I_h) and (I_f) must certainly cover either $l(I_g)$ or $r(I_g)$. If one of them covers only $l(I_g)$ and the other covers only $r(I_g)$ then Observation 3.2.3

39

Procedure Sweep;

{

    1 Initialize $S_1 \leftarrow b_u, h_u$ and $S_2 \leftarrow b_d, h_d$;

    2 $v^* :=$ closest vertical edge to L lying on its right such that it is either adjacent to one of b_u^*, b_d^*, h_u^* and h_d^* in P, or the Y-interval it spans, say $I_y(v)$ is contained in (y_{h_u}, y_{b_u}) or contained in (y_{b_d}, y_{h_d});

    3 Update L to contain v^*;

    4 if $(v^*$ is adjacent to both h_u^* and $h_d^*)$ then STOP;

        % This will happen at least at the last vertical hole edge between b_u^* and b_d^* encountered by L during the sweep—also when this happens b_u^* and b_d^* can see each other along a vertical line immediately to the right of v^* %

    5 elseif v^* is adjacent to $b_u^*(b_d^*)$ (or $h_d^*(h_u^*))$ {   % See Figure 3.5a (or b) %

        $e^* =$ the other horizontal edge adjacent to v^*;

        if $(e^*$ is not to the right of $v^*)$ then

            $e^* =$ edge of P crossing L immediately above(below) $b_u^*(b_d^*)$ (or $h_d^*(h_u^*))$;

        $b_u(b_d)$ (or $h_u(h_d)) = e$;

        }

    6 else
    % In this case $[I_y(v)] \subset [y_{b_u}, y_{h_u}]$ ($[I_y(v)] \subset [y_{b_d}, y_{h_d}]$) and both the edges adjacent to v^* are to its right (Figure 3.7) %

        if $(v^*$ is on the outer boundary of $P)$ then
            $b_u^*(b_d^*) = e^* =$ lower(upper) of the two edges adjacent to v^*;

        else

            $h_u^*(h_d^*) = e^* =$ upper(lower) of the two edges adjacent to v^*;

    7 if $(e^*$ is the new $b_u^*(b_d^*)$ or the new $h_u^*(h_d^*))$ then

            if (either both e^* and the last entry of $S_1(S_2)$ are hole edges or both are edges of the outer boundary) then
                replace the last entry of $S_1(S_2)$ by e;

                    % In this case if e' is the last-but-one element of $S_1(S_2)$ and e'' is the one preceding it then $e''e'e$ is a simple path of G %

            else append q to $S_1(S_2)$;

                    % here if e'' and e' are the last two elements of $S_1(S_2)$ then again $e''e'e$ is a simple path of G %

    8 go to Step 2;

}

Figure 3.6: Procedure Sweep
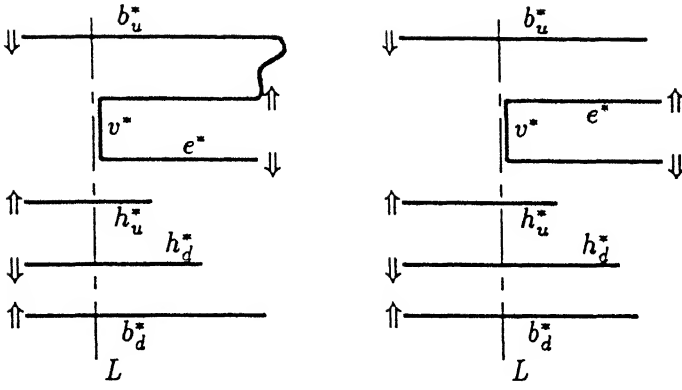
Figure 3.7: Sweep update when $v^* \subset (b_u, h_u)$ is (a) on the outer boundary of P (b) a hole edge of P.
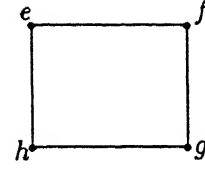
Figure 3.8: The four-cycle $efgh$ of G.

(a)

(b)
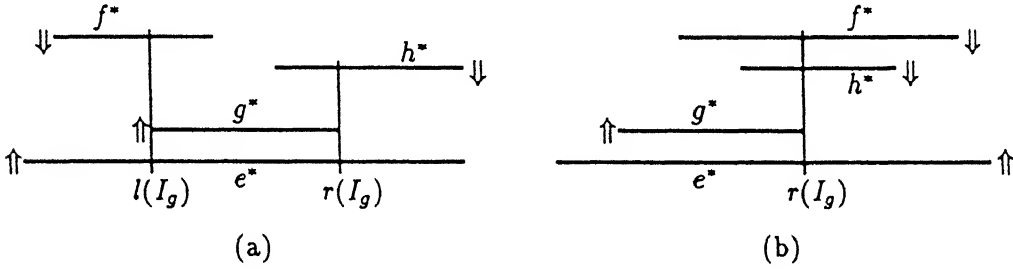
Figure 3.9: Existence of a leaf adjacent to (a) e (b) f.

tells us that e does have a leaf adjacent to it in G (see Figure 3.9a). If however both cover some end of $[I_g]$ (say the right end) then certainly $[I_h] \subset (I_f)$ (see Figure 3.9b). Now again Observation 3.2.3 tells us that f must necessarily have a leaf adjacent to it in G.

case 2: $[I_g] \not\subset [I_e]$

Without loss of generality assume that $r(I_g) < r(I_e)$ Then certainly to see both e^* and f^*, both (I_h) and (I_f) must cover $r(I_g)$. Now the argument in case 1 again tells us that f must have a leaf adjacent to it in G.

Hence the claim. ∎

We may mention here that it is not necessary for every four-cycle to have a 'distinct'

41
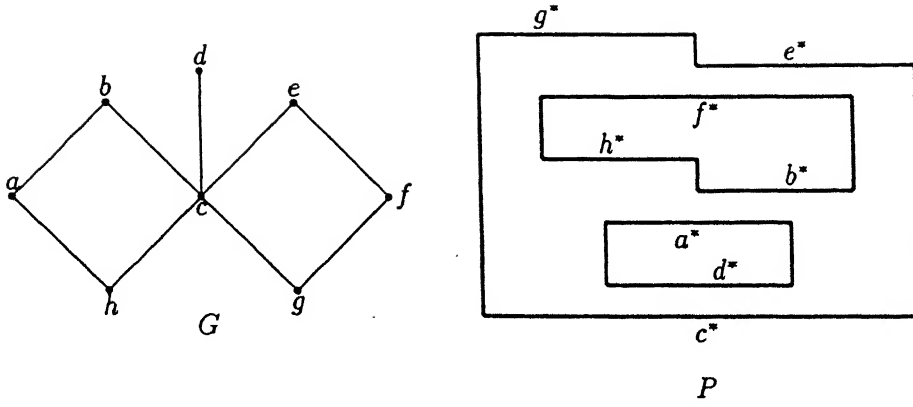
Figure 3.10: P realizes G which has two four-cycles 'sharing' a leaf.

leaf adjacent to one of its vertices. It is possible for a graph with several four-cycles 'sharing' a leaf, to be realizable as shown Figure 3.10. Note that the construction in Figure 3.10 can be extended to realize an arbitrarily large number of four-cycles sharing a single leaf.

Necessary Condition O6: *Let G be a graph with cycles (not a tree) and let $G = G_0, G_1, G_2, \ldots, G_k$ be a sequence of graphs where $G_i \neq G_{i-1}$ is obtained by removing every leaf e of G_{i-1} such that either*

1. *in G_{i-1}, f, the vertex adjacent to e has exactly one non-leaf vertex adjacent to it, or*

2. *e is not a leaf of G_{i-2},*

for $1 \leq i \leq k$. Then there must exist a polygon P_k which realizes G_k. Graphs for which $\max(k) = 0$ will henceforth be referred to as irreducible graphs.

Proof: We show by induction on i that there exists a polygon P_i realizing G_i for every i, $1 \leq i \leq k$. The base instance is trivially the polygon P realizing G.

To construct P_1 from P we first note that since $i = 1$, condition 2 under which a leaf of G_0 can be deleted, does not apply. Therefore consider a vertex f of G satisfying the first condition. Without loss of generality let f^* face upward and let $e_1^*, e_2^*, \ldots, e_l^*$ be the edges of P corresponding to the leaves $e_1, e_2, \ldots, e_l$ of G adjacent to f. We'll now show

42

how the edges $e_i^*, 1 \leq i \leq l$ can be pruned from P to obtain a polygon whose visibility graph is $G - \{e_1, e_2, \ldots, e_l\}$. Since each e_i, $1 \leq i \leq l$ is a leaf, clearly $[I_{e_i}] \subset (I_f)$. Moreover from Observation 3.2.2 it also follows that the rectangle bounded by e_i^*, f^*, $x = r(I_{e_i})$ and $x = l(I_{e_i})$ is empty. Suppose $(I_{e_i} \cap I_{e_j}) \neq \phi$ for some $i \neq j$ and let $y_{e_i} > y_{e_j}$. Then again from Observation 3.2.2 it is clear that e_i is not a leaf because e_i^* must certainly see some edge other than f^* over the interval $[I_{e_j} \cap I_{e_i}]$. This contradicts our assumption. The intervals (I_{e_i}), $1 \leq i \leq l$ are therefore pairwise disjoint. The e_i^*'s can now be ordered linearly in the X-direction—say e_i^* lies to the left of e_{i+1}^*, $1 \leq i \leq l$ without loss of generality.

Suppose for some i, e_i^* and e_{i+1}^* are not consecutive edges on P. Then clearly $r(I_{e_i}) < l(I_{e_{i+1}})$. Thus since f has exactly one non-leaf vertex adjacent to it, there is exactly one edge e^* which can see f^* in the interval $[r(I_{e_i}), l(I_{e_{i+1}})]$. Obviously e is not a leaf. However it is not necessary that the edges visible to f^* through the 'gaps' between two pairs of non-consecutive edges (e_i^*, e_{i+1}^*) and (e_j^*, e_{j+1}^*) be distinct. We now have two cases.

case 1: There is at most one pair of non-consecutive edges (e_i^*, e_{i+1}^*).

case 1.1: There are no pairs of non-consecutive edges.

case 1.1.1: $r(I_{e_l}) = r(I_f)$.

Then obviously because P is in general position, e_l^* and f^* would be consecutive on P with a common vertical edge v_l^* at $x = r(I_f)$ adjacent to both. The edges e_i^*, $1 \leq i \leq l$ can now be pruned without changing the rest of the polygon by extending the vertical edge at $x = l(e_1^*)$ downward to meet f^*. See Figure 3.11. Note here that because e_1^* is a leaf $l(I_{e_1}) > l(I_f)$. A similar pruning can be carried out also when $l(I_{e_1}) = l(I_f)$.

case 1.1.2: $r(I_{e_l}) \neq r(I_f)$ and $l(I_{e_1}) \neq l(I_f)$.

Again because both e_1 and e_l are leaves, certainly $r(I_{e_l}), l(I_{e_1}) \in (I_f)$. Let e_0^* and e_{l+1}^* be the edges other than e_{l-1}^* and e_2^* which are consecutive to e_1^* and e_l^* respectively in P. It is easy to infer from Condition 1 that neither e_0^* nor e_{l+1}^* can see f^*. Moreover e_0^*
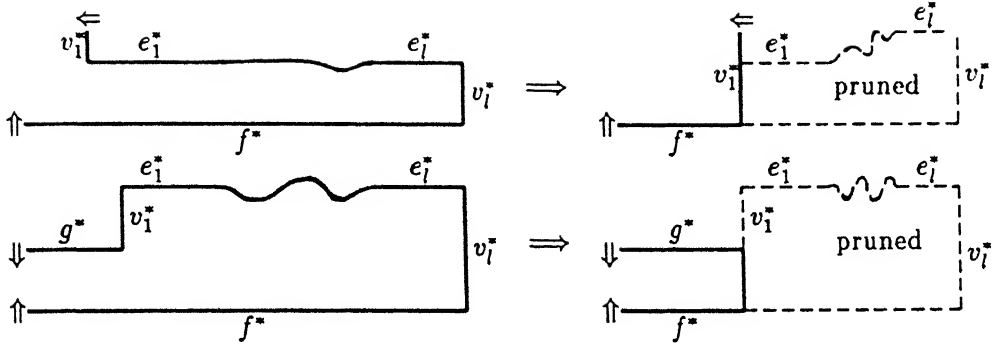
Figure 3.11: 'Pruning' the leaves $e_1, \ldots, e_l$ when $r(I_{e_l}) = r(I_f)$

and e_{l+1}^* must be above e_1^* and e_l^* respectively. So let g_l^* and g_1^* be respectively the edges of P closest to and above e_l^* and e_1^* such that $r(I_{e_l}) \in (I_{g_l})$ and $l(I_{e_1}) \in (I_{g_1})$. Since P is in general position g_1^* and g_l^* must exist. From Observation 3.2.2 it is clear that neither is a leaf. Hence again according to Condition 1 of the statement of the lemma, $g^* = g_1^* = g_l^*$. This implies in turn that $\cup_{i=1}^l [I_{e_i}] \subset (I_g)$. One can now easily see that the e_i^*'s are not in the same connected component of P as either g^* or f^*. Moreover since g is the only vertex of G other than the e_i's adjacent to f, $[I_f] \subset [I_g]$.

If $r(I_g) = r(I_f)$ then g^* and f^* are consecutive on P with the vertical edge v^* at $x = r(I_g)$ adjacent to both. The pruning can now be done in two stages. In the first stage we reduce P to another polygon to which the pruning strategy shown in Figure 3.11 can be applied. For this we first extend e_l^* to meet v^* on the right and v_l^* up to meet g^*. Later we remove v_l^* and the portions of the edges g^* and v^* between $r(I_{e_l})$ and $r(I_g)$ and y_g and y_{e_l} respectively. See Figure 3.12. Note that the rectangle bounded by $x = r(I_{e_l})$, $y = y_{e_l}$, v^* and g^* is empty. The reduction therefore either merges two holes or a hole with the outer boundary depending on whether g^* is a hole edge or not. In any case one can easily verify that the visibility graph of P remains unchanged with this transformation. It is also clear that the pruning of Figure 3.11 can now be applied to the transformed P.

So let $r(I_g) \neq r(I_f)$. We in any case know that the rectangle bounded by $x = r(I_{e_l})$,

44
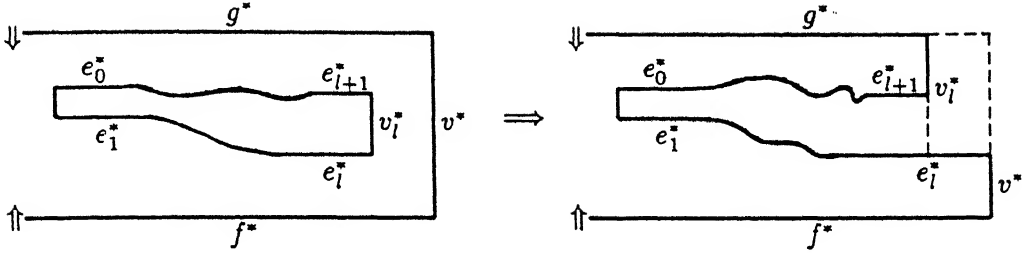
Figure 3.12: Reduction of P for pruning when $r(I_g) = r(I_f)$
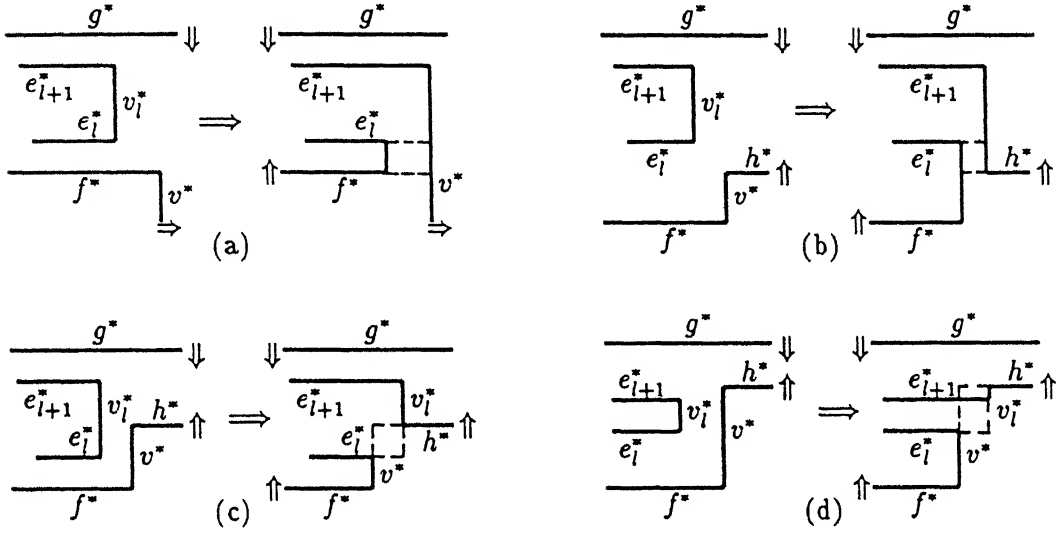
Figure 3.13: The reduction when $r(I_g) \neq r(I_f)$.

g^*, $x = r(I_f)$ and f^* is empty. So since both the end vertices of v_l^* are concave, we 'pull' v_l^* to the right upto $x = r(I_f)$. Note that f^* can still see g^* over the interval $[l(I_f), l(I_{e_1})]$. The subsequent reduction to the case where pruning as in Figure 3.11 can be applied is illustrated in Figure 3.13 when (a) the vertical edge v^* adjacent to f^* at $r(I_f)$ is below f^*, (b) v^* is above f^* with the upper end vertex below y_{e_l} (c) the upper end vertex of v^* is above y_{e_l} but below $y_{e_{l+1}}$ and (d) the upper end vertex is above $y_{e_{l+1}}$. Each of these reductions clearly preserve the visibility structure of P.

case 1.2: There is exactly one pair of non-consecutive edges (e_i^*, e_{i+1}^*).

45
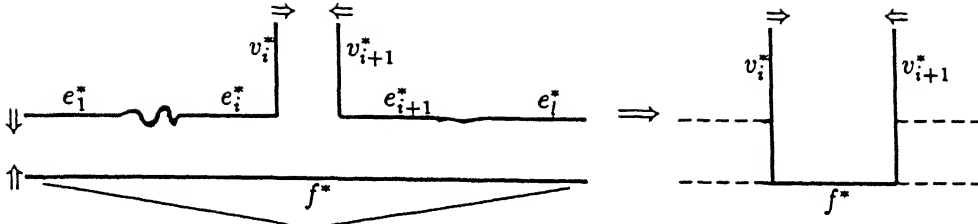
The two ends are reduced as in Figure 3.13.

Figure 3.14: The reduction when there is exactly one pair (e_i^*, e_{i+1}^*) of non-consecutive edges.
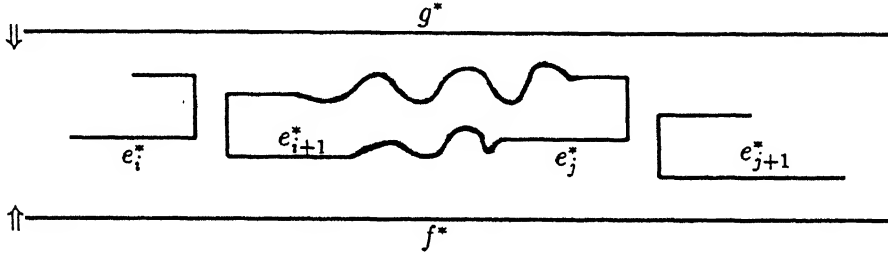
Figure 3.15: More than one non-consecutive pairs of edges in P among $e_i^*, 1 \leq i \leq l$.

Clearly the transformations shown in Figure 3.12 and 3.13 can be used to reduce the left and right 'ends' of the portion of P shown in Figure 3.14 so that the sets of leaves $\{e_1^*, \ldots, e_i^*\}$ and $\{e_{i+1}^*, \ldots, e_l^*\}$ can be pruned separately as in Figure 3.11 by extending v_i^* and v_{i+1}^* down to meet f^*.

case 2: There are more than one pairs of non-consecutive edges in the sequence of e_i's.

As observed earlier, for condition (i) to apply, the edges visible to f^* through each of the 'gaps' between non-consecutive edges must all be the same. Let this edge be g^*. Let (e_i^*, e_{i+1}^*) and (e_j^*, e_{j+1}^*) be two pairs of non-consecutive edges so that $e_{i+1}^*, e_{i+2}^*, \ldots, e_j^*$ are consecutive (assume without loss of generality that $j > i$). Clearly $e_{i+1}^*, e_{i+2}^*, \ldots, e_j^*$ must all belong to a single hole of P which contains none of e_i^*, e_{j+1}^*, g^* and f^*. See Figure 3.15. Here we first reduce the problem to that in case 1.2 from which the rest of the reductions and pruning will follow.

46

For this the hole containing $e_{i+1}^*, e_{i+2}^*, \ldots, e_j^*$ can be merged with the component of P containing e_i^*, again without affecting the visibility graph of P. The merging strategy is exactly like that shown in Figure 3.13. Every such merger thus reduces the number of such pairs of non-consecutive edges by one. The polygon P can therefore be reduced to one to which the reductions implied by case 1.2 can be applied.

We have thus shown how P_1 realizing G_1 can be obtained from P. Note that the polygon P_1 is also in general position—our reductions ensure that. Moreover after $e_1^*, e_2^*, \ldots, e_l^*$ are pruned from P, f^* sees exactly one edge of P_1. It may be noticed here that in each of our reductions f^* becomes an edge of P_1 at least one of whose end-vertices (polygon) is convex. Thus taking the transformation from P to P_1 as the basis, we incorporate this (that the polygon edge corresponding to a leaf covered by Condition 2 has at least one convex end vertex) too into our induction hypothesis.

To carry on with the induction, suppose we have obtained P_i realizing G_i. The above discussion already tells us how the polygon edges of P_i corresponding to the leaves of G_i covered under Condition 1 can be pruned. So all that remains to be seen is how the portion of P_i corresponding to a leaf e covered by Condition 2 is to be pruned from P_i. Our modified induction hypothesis tells us that at least one end vertex of e^* is convex in P_i. Note that since e was not a leaf of G_{i-1}, the leaves adjacent to e must have been pruned according to condition 1 while obtaining G_i from G_{i-1}. Let f be the vertex of G_i adjacent to e. Assume without loss of generality that f^* faces upward in P_i. Because e is a leaf clearly $[I_e] \subset [I_f]$.

case A: $r(I_e) = r(I_f)$.

Then e^* and f^* must be consecutive on P_i. The pruning of e^* in this case by extending v^* (the vertical edge adjacent to e^* at its left end) down to meet f^* is done just as in Figure 3.11.

The same holds even when $l(I_e) = l(I_f)$.
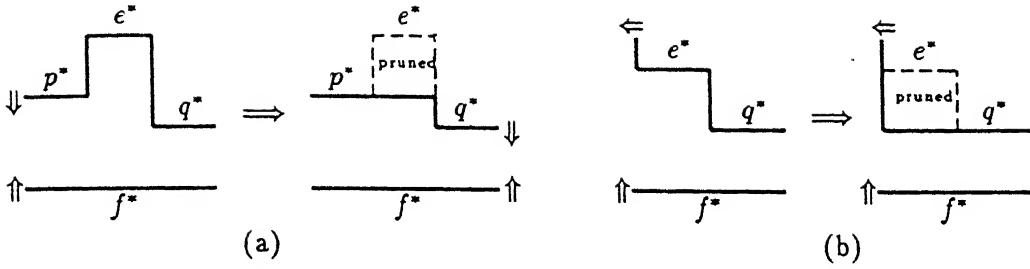
Figure 3.16: Pruning a leaf under condition 2.

case B: $[I_e] \subset (I_f)$.

If both the end vertices of e^* in P_i are convex, then e^* is pruned by extending p^* to the right to meet the vertical edge between e^* and q^* (p^*, e^* and q^* are consecutive on P_i). See Figure 3.16a. Remember that since e^* is a leaf, the rectangle bounded by $x = l(I_e)$, e^*, $x = r(I_e)$ and f^* is empty.

If however one of the end vertices is concave, then the vertical edge between p^* and e^* is extended down to meet q^* extended to the left as shown n Figure 3.16b to prune e^*.

This settles the induction step thus implying that G_k is indeed realizable. $\blacksquare$

3.3 Towards Sufficiency

In this section we give some conditions that are *sufficient* for a graph to be realizable. We also show a non-realizable graph which however satisfies Necessary Conditions O1-O6. Thus our set of necessary conditions is not sufficient for a graph to be realizable.

Lemma 3.3.1: *Let G be the vertical edge visibility graph of a polygon P. If G' is a graph obtained by adding trees to the non-leaf vertices of G, then G' too is realizable.*

Proof: Let e^* be the edge of P corresponding to a non-leaf vertex e of G. Suppose f^* is the closest visible edge of P to e^*. It is then easy to show that since e is not a leaf,

48
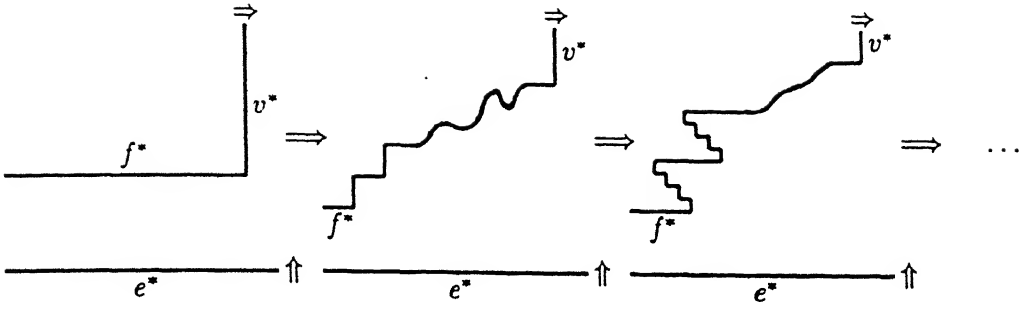
Figure 3.17: Augmenting the visibility graph with an arbitrary tree at a non-leaf vertex.

either $l(I_f) \in (I_e)$ or $r(I_f) \in (I_e)$ or both. Let $r(I_f) \in (I_e)$ without loss of generality. We now show how one can obtain a polygon with its visibility graph being G augmented by an arbitrary tree rooted at e. We account for the new tree at the right corner of f^*. The strategy is exactly the one used in [36, Lemma 7.4]. This is illustrated in Figure 3.17. For instance if the root of the extra tree (the root is e) is of degree k, then we 'truncate' the right corner of f^* with a 'staircase' of k steps. For each of these level-one edges in the staircase, we build level-two staircases, each with as many steps as the degree of the parent level-one vertex minus one and so on. This can be clearly carried on indefinitely till the whole tree is realized. Also note that this can be done independently for each non-leaf vertex of G without interferences with the visibilities of the other edges. Thus G' too is realizable. Hence the lemma. ∎

The above lemma along with Necessary Condition O4 which we proved earlier in fact completely characterizes the trees which are realizable. We thus have the following theorem.

Theorem 3.1: *A tree G is realizable if and only if it has a pair of leaves separated by a distance of three on G.*

Proof: The theorem follows from the fact that any such tree can be built up from a simple path of length three by adding subtrees to non-leaf vertices. ∎

49
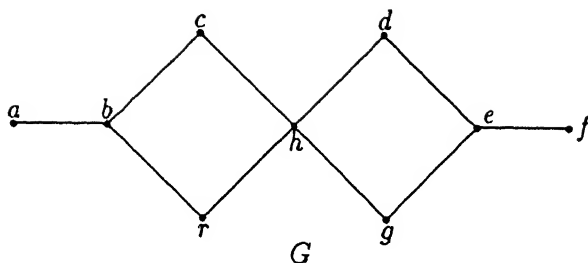
Figure 3.18: An unrealizable graph which satisfies all the six necessary conditions.

As for graphs with cycles we will first show through 'ad-hoc' arguments that our set of necessary conditions are not sufficient for a graph to be realizable. Our counterexample is shown in Figure 3.18.

Theorem 3.2: *The Necessary Conditions O1-O6 are not sufficient.*

Proof: Suppose the graph G of Figure 3.18 is realizable. In a polygon P realizing G let b^* be facing upward. Then clearly c^* and r^* face downward, h^* faces upward and $y_e, y_r > y_b, y_h$. Suppose for some $p \in \{b, h\}$, $y_p < y_q$ for $q = \{b, h\} - p$. From the arguments in the proof of Necessary Condition O5 it is clear that neither of $[I_c]$ and $[I_r]$ can contain the other because otherwise the 'larger' of the two must have a leaf adjacent to it in G. Now the only possibility left is where $[I_q] \subset (I_p)$ with $r(I_q) \in (I_{p'})$ and $l(I_q) \in (I_{q'})$ for $\{p', q'\} = \{c, r\}$. In this case p must certainly have a leaf adjacent to it. Thus $p = b$ and $q = h$.

An identical argument will tell us that e^* must be below h^* with $[I_h] \subset (I_e)$. But it cannot be that $y_h > y_e > y_b$ because in that case c^*, r^* and b^* will certainly have extra visibilities according to Observation 3.2.2. Thus $y_e < y_b$. But now reversing our preceding argument we see that e, d and g must have edges not in G. Thus it is impossible to obtain a realization of G. $\blacksquare$

In the rest of this chapter we will show that Necessary Conditions O1-O6 are sufficient for a graph to be an orthogonal edge visibility graph *upto leaf addition*. More precisely,

we show that from any connected bipartite planar graph G we can obtain a realizable graph G' such that every vertex in $V(G') - V(G)$ is a leaf of G'. We first prove this claim for irreducible graphs. The full claim will then follow easily from Lemma 3.3.1.

Lemma 3.3.2: *If G is a connected bipartite planar irreducible graph then there exists a realizable graph G' such that $|V(G') - V(G)| \le F(G)$ with every vertex of $V(G') - V(G)$ being a leaf of G'. $F(G)$ is the number of bounded faces in a planar embedding of G.*

Proof: Let us assume initially that G is two connected. We will later show how this can be extended to singly connected graphs as well.

Consider any planar embedding $M(G)$ of G. Let $C_1, C_2, \ldots, C_r$ be a sequence of Jordan curves on the plane such that C_1 bounds a region adjacent to the unbounded region of $M(G)$ (the regions of $M(G)$ are the connected components of $\Pi - M(G)$, Π being the plane) and the number of connected components of $\Pi - M(G_i)$ is exactly one more than the number of components of $\Pi - M(G_{i-1})$, $1 \le i \le r$, where G_i is the subgraph of G induced on the set of vertices e for which $M(e) \in \Gamma C_i$. This actually corresponds to an ordering of the bounded regions of $M(G)$ such that the closure of the union of any prefix of that sequence is a simply connected region with a Jordan curve (a C_i) as its boundary. That an ordering of the kind described above exists is easily seen from the facts that every bounded region of $M(G)$ is simply connected if G is connected and is bounded by a Jordan curve if G is 2-connected. Clearly $M^{-1}(C_i)$ is a simple even length cycle of G for $1 \le i \le r$. Remember that G being bipartite, has only even length cycles. Also $M^{-1}(\Gamma(C_i - C_{i+1}))$ is a simple path S_i of G for $1 \le i < r$, where Γ gives the 'closure' including the end-points of the curve $C_i - C_{i+1}$. Note that for all $1 \le i \le r$, G_i too is two-connected. Thus S_i must contain at least two vertices of G_i, $1 \le i \le r$. We now prove our claim through induction that for any $1 \le i \le r$, there exists a polygon P_i with visibility graph G'_i such that every vertex of $V(G'_i) - V(G_i)$ is a leaf of G'_i and $|V(G'_i) - V(G_i)|$ is at most i. We in fact show something stronger—that there exists such a polygon P_i for which

51

1. the outer boundary is vertically convex i.e., any vertical line intersects the outer boundary at exactly two points,

2. $f_0^*, f_1^*, \ldots, f_{k_i+1}^*, f_0^*$ is the outer boundary (in clockwise order), f_0^* being the only downward facing edge, and

3. the vertices of G_i in their order of occurrence on C_i are

$$f_0, f_1, e_1, f_2, e_2, \ldots, e_{k_i}, f_{k_i+1}, f_0$$

where e_j^* is a downward facing hole edge for $1 \leq j \leq k_i$.

We first observe a consequence of the outer boundary of P_i being vertically convex, which is crucial to the rest of our proof. It is that every edge f_j^* can be placed independently at any arbitrary ordinate—the only constraint being that none of them can intersect a hole of P_i—without changing the visibility graph of the polygon in any way. Thus f_0^* being the only downward facing edge, it follows that each f_j^* , for $1 \leq j \leq k_i + 1$ can be 'pulled down' as far as we please independent of the rest.

We are now ready to carry on with the proof. Our base instance is G as a simple even length cycle. A polygon having a single hole with a downward facing staircase and a matching upward facing staircase on the outer boundary as shown in Figure 3.19 does satisfy the induction hypothesis, as can be easily verified.

Suppose now that the claim holds for any $j \leq i$ with a polygon P_j. We will now construct P_{i+1} realizing G_{i+1} upto at most $i + 1$ leaves. We consider the several cases that arise separately below and in each of the cases it is easy to verify that the conditions of the induction hypothesis are indeed preserved. Moreover in each of the cases we assume that G_{i+1} has at most two vertices more than G_i and give a 'basis' construction under this assumption. It can again be seen easily that this basis construction can be extended to cases in which $|V(G_{i+1}) - V(G_i)|$ (or equivalently $|M^{-1}(C_{i+1} - C_i)|$) is arbitrarily large by replacing the two portions (one on a hole and the other on the outer boundary) of the
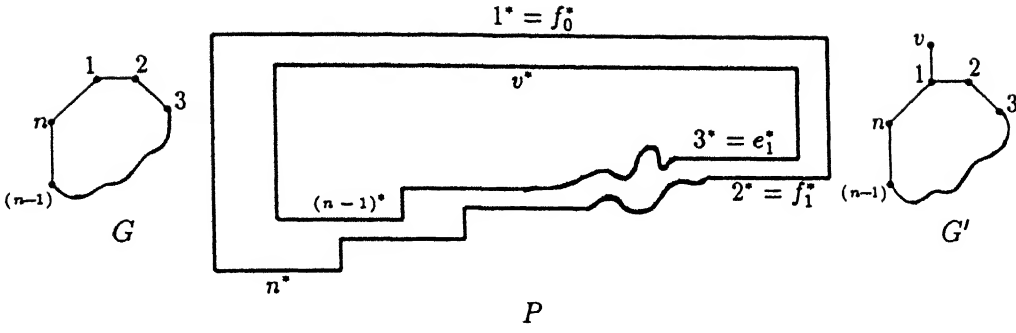
Figure 3.19: Realization of a simple even length cycle.

polygon P_{i+1} for the base case, shown shaded ($/\!/\!/$) in the figures we use for illustration, by a pair of matching staircases of appropriate length. Let the vertical edge of the outer boundary of P_i between f_j^* and f_{j+1}^* be v_j^* and let a and b denote the end vertices of S_i.

case 1: S_i is an even length path of G_i.

Then either both a^* and b^* face upward or both face downward in P_i. Moreover in this case $|V(G_{i+1}) - V(G_i)| = 1$. So let $\{g\} = V(G_{i+1}) - V(G_i)$.

case 1.1: both a^* and b^* face upward.

From our induction hypothesis clearly both a^* and b^* belong to the outer boundary of P_i. Without loss of generality let a^* be to the right of b^* (note that because the outer boundary is vertically convex, $(I_a \cap I_b) = \phi$). So let $a = f_s$ and $b = f_t$, with $t > s$.

If $t = s + 1$ then we add a new rectangular hole cutting the segment joining the upper end vertex of v_s^* to e_s^* so that the downward facing edge of the new hole is g^*. See Figure 3.20a. Otherwise we first exploit the observation we made at the beginning and 'pull' a^* and b^* downwards far enough to 'clear' the rest of the polygon. Then we create a new hole consisting of the chain of the outer boundary of P_i clockwise from v_s^* to v_{t-1}^* along with a horizontal edge g^* joining the two. Finally f_t^* and f_s^* are made consecutive on the outer boundary with a single step. See Figure 3.20b.

case 1.2: Both a^* and b^* face downward.
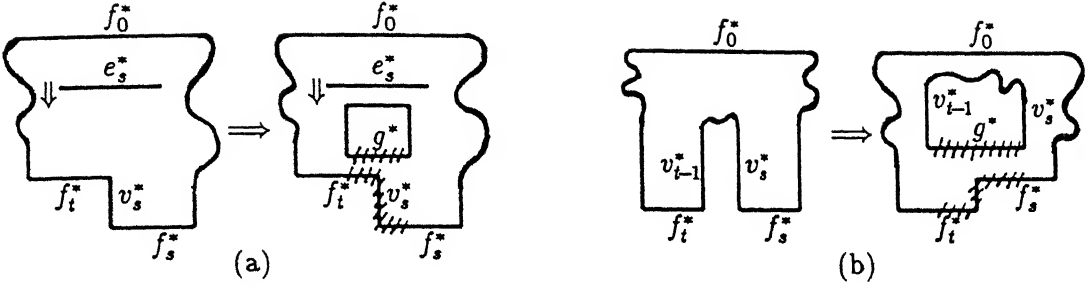
Figure 3.20: Obtaining P_{i+1} from P_i when $a = f_s$ and $b = f_t$ with $t = s+1$.
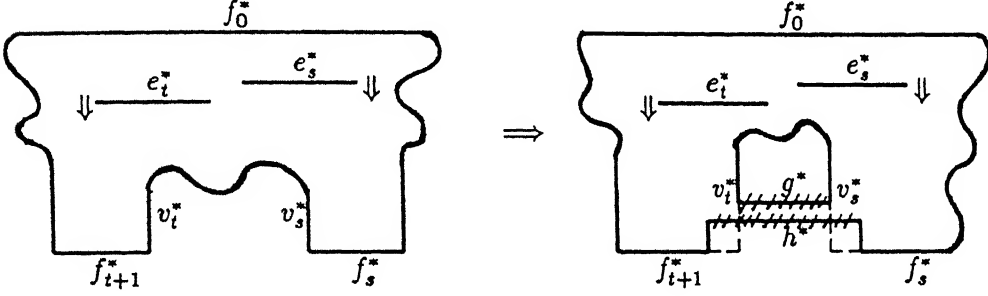
Figure 3.21: Obtaining P_{i+1} from P_i when $a = e_s$ and $b = e_t$ with $t > s$.

The induction hypothesis implies that either both are hole edges or one of them is f_0^* and the other is a hole edge.

case 1.2.1: Both are hole edges of P_i.

Let a^* and b^* be respectively e_s^* and e_t^* with $t > s$. We obtain P_{i+1} by first creating a new hole containing the clockwise portion of the outer boundary of P_i from v_s^* to v_t^* and a new horizontal edge g^* joining the two. Then we introduce a new edge h^* on the outerboundary such that f_s^*, h^* and f_t^* are consecutive with h^* visible to both e_t^* and e_s^*. See Figure 3.21.

case 1.2.2: $a = f_0$ and $b = e_t$.

Suppose that $M(f_{k_i+1}) \notin C_{i+1}$. Then we create a new hole consisting of the clockwise chain of the outer boundary from v_t^* to $f_{k_i+1}^*$ along with a vertical edge adjacent to and below $f_{k_i+1}^*$ and a horizontal edge g^* adjacent to and on the left of v_t^*. After that a
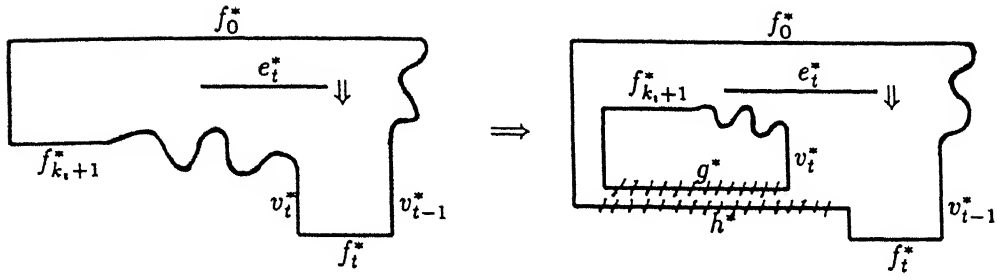
Figure 3.22: Obtaining P_{i+1} from P_i when $a = f_0$ and $b = e_t$.

new edge h^* is added to the outerboundary such that f_t^*, h^* and f_0^* are consecutive on the outer boundary with h^* seeing both f_0^* and e_t^*. See Figure 3.22. The construction for the case where $M(f_{k_i+1}) \in C_{i+1}$ would be identical except that we start from f_{t+1}^* and proceed with the above construction on its right. Note that now $M(f_1) \notin C_{i+1}$.

case 2: S_i is an odd length path.

In this case one of a^* and b^* (say b^*) faces upward and the other (a^*) faces downward. Note that from our induction hypothesis, $b = f_t$ for some t, $1 \leq t \leq k_i + 1$ and a is either some $e_s, 1 \leq s \leq k_i$ or f_0. Also $V(G_{i+1}) - V(G_i) = \phi$ i.e., just an extra edge is added between a and b to get G_{i+1}, unless $a = f_0$ or e_t. In the latter two cases we will have $|V(G_{i+1}) - V(G_i)| = 2$ because a^* can already see b^* in P_i.

case 2.1: $a = f_0$.

Here if $t \neq k_i + 1$ then the construction is similar to that for case 1.2.2 as shown in Figure 3.22 except that we extend f_t^* itself to the left so that it sees f_0^* instead of introducing a new edge h^* between the two. If however $t = k_i + 1$ then a rectangular hole added on the left of all the already existing holes would not change the visibility graph of P_i apart from adding two leaves, the edge g^* of P_i (facing downward) corresponding to one of which will eventually be replaced by a staircase matching with one on $f_{k_i+1}^*$.

case 2.2: $a = e_s$.

Suppose $s = t$. So let $\{g, h\} = V(G_{i+1}) - V(G_i)$. In this case we introduce a new

55
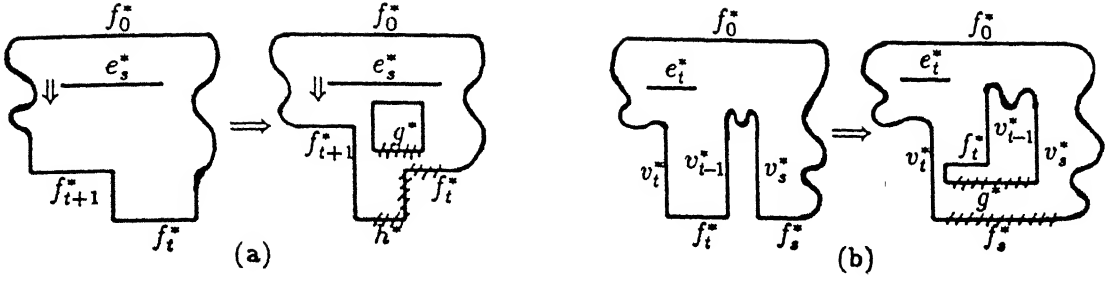
Figure 3.23: Obtaining P_{i+1} from P_i when S_i is an odd length path.

horizontal edge h^* between f_t^* and f_{t+1}^* so that it sees e_t^* and a new rectangular hole cutting the segment joining the upper end vertex of v_t^* with e_t^* the downward facing edge of which is g^*. See Figure 3.23a.

We know that if neither one of cases 2.1–2.2 is applicable, then $V(G_{i+1}) = V(G_i)$. If $s < t$ then our construction is identical to the one shown in Figure 3.20b except that here f_t^* must see e_s^* in the new polygon. If however $s > t$ then we first pull both f_s^* and f_t^* down far enough to clear the rest of the polygon with f_t^* reaching out a little below f_s^*. Then we create a new hole consisting of the clockwise portion of the outer boundary from v_t^* to f_s^*, a new vertical edge below f_s^* and a new horizontal edge joining this with v_t^*. Finally we extend f_t^* to the left to meet v_s^* (so that it sees e_s^*). See Figure 3.23b.

This exhausts all the cases hence proving the lemma by induction for two-connected graphs. Note that the leaves of G if not already in the visibility graph of P can be easily incorporated later by invoking the procedure in Lemma 3.3.1. We also note that the polygon produced as described above is wholly below f_0^* and confined to the region bounded by $x = l(I_{f_0})$ and $x = r(I_{f_0})$. This property of our construction will come in very handy when we attempt to extend our above reconstruction scheme to singly-connected graphs.

Let H_i, $1 \leq i \leq k$, be the two-connected components and v_i, $1 \leq i \leq s$, the cut-vertices of G. Consider a graph G_c formed from G consisting of a vertex h_i for every H_i, $1 \leq i \leq k$ and a vertex u_i for every v_i, $1 \leq i \leq s$ with $(h_i, u_j) \in E(G_c)$ only when $v_j \in V(H_i)$. It is easy to show that G_c is in fact a tree.

56

Let h_t be a distinguished vertex of G_c. For every other vertex h_i of G_c, let u_{t_i} be the vertex adjacent to h_i on the path from h_t to h_i in G_c. We'll now see how G_c can be used to obtain a realization of G upto leaves.

First for every two-connected component H_i of G, we construct a polygon Q_i realizing H_i as described in the earlier part of the proof of the lemma. Also for $i \neq t$, Q_i is such that the only downward facing polygon edge f_{0i}^* on the outer boundary of Q_i corresponds to the vertex v_{t_i} of G. We next describe how the Q_i's can be put together to realize G.

Consider a cut-vertex $v \in V(H_t)$. Since H_t is two-connected clearly either v is not a leaf of H_t or H_t consists of only two vertices including v and an edge between them. Thus if v is leaf then obviously Q_t is just a rectangle. If however v is not a leaf then let u^* be the closest edge visible to v^* in Q_t. In this case it is easy to see that either $r(I_u) \in (I_v)$ or $l(I_u) \in (I_v)$. We assume without loss of generality that $r(I_u) \in (I_v)$ and that v^* faces upward in Q_t.

We illustrate our construction for putting the polygons for the two-connected components together, below, with the assumption that the right end-vertex of u^* is concave. The construction however is equally valid even if the right end-vertex is convex with possibly $r(I_u) = r(I_v)$. Note that construction is thus valid even if v is a leaf with u being the vertex of H_t.

Let $H_{m_1}, H_{m_2}, \ldots, H_{m_l}$ be the two-connected components of G apart from H_t containing v. We now replace the right corner of u^* by a staircase with l steps, the ith step consisting of the vertical edge p_i^* and the horizontal edge q_i^*, $1 \leq i \leq l$ (see Figure 3.24) i^{th} step with Now for every i, $1 \leq i \leq l$, the polygon Q_{m_i} is 'shrunk' so that the length of $f_{0m_i}^*$ is exactly that of q_i^* and that of the vertical edge of Q_{m_i} at its right end is less than that of p_{i+1}^*. Each Q_{m_i} is then placed so that $f_{0m_i}^*$ coincides with q_i^* and Q_{m_i} is above $f_{0m_i}^*$ after which $f_{0m_i}^*$ and the vertical edge at its right end vertex are 'removed'. The polygon thus obtained is clearly a realization of $H_t \cup \cup_{i=1}^l H_{m_i}$ upto leaves. Note that the only leaves in the visibility graph of the polygon thus obtained but not in $H_t \cup \cup_{i=1}^l H_{m_i}$ are the
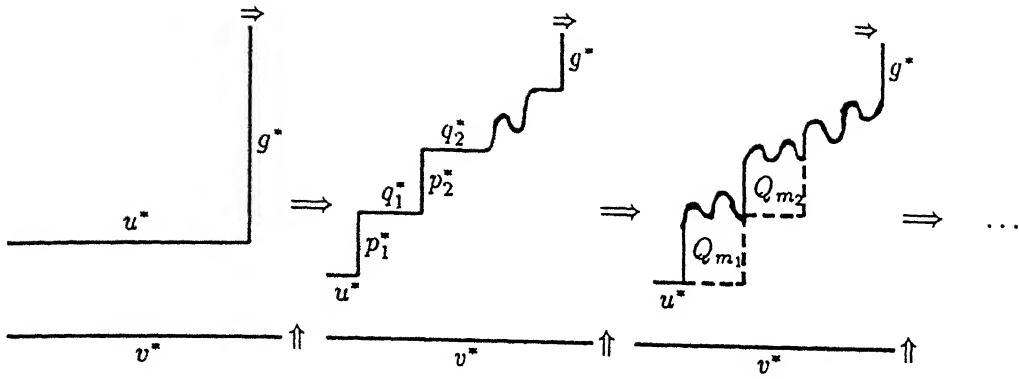
Figure 3.24: Putting the polygons for the two-connected components of G together

ones already in the H_{m_i}'s i.e., putting the polygons for the two-connected components of G together does not introduce any more leaves into the visibility graph. Finally it is easy to see that this can be carried out progressively through the whole of G_c to finally realize the G upto leaves as in the statement of the lemma. ∎

Chapter 4

Orthogonal Edge Visibility

Graphs — II

In this chapter we address the problem of the realizability of a pair of trees with the same number of nodes in each. Here we construct two *fairly large* classes of pairs of trees which are not jointly realizable. The only known results on this problem are the ones by O'Rourke [36, chapter 7].

The rest of the chapter is organised as follows. In section 1, we introduce the preliminary definitions and some notation. In section 2, we briefly discuss the results of [36] and also make some additional observations. We construct our classes of unrealizable pairs in sections 3 and 4. We also show examples to assert that these two classes are in some sense independent or in other words neither class is a subclass of the other. Finally in section 5, we end with some concluding remarks.

59
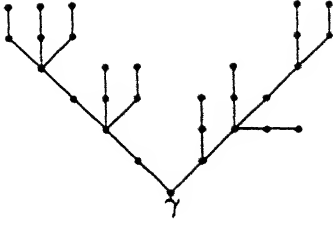
Figure 4.1: A tree in $L_2(3)$.
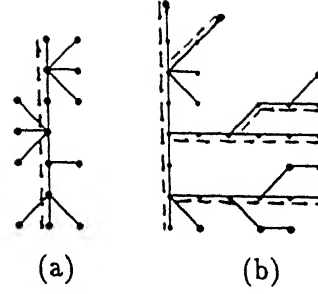
Figure 4.2: (a) a caterpillar and (b) a 2-caterpillar. The spines at all the levels are shown as broken lines.

4.1 Definitions and Notation

All of the terminology introduced in this section is for an *undirected tree* $T = (V, E)$ unless otherwise mentioned. Let $T_1, T_2, \ldots, T_t$ be the components obtained after a node $v \in V$ is removed. The induced subtrees on $V(T_i) \cup \{v\}, 1 \leq i \leq t$, are called the *subtrees* of T at v. The distance $d(u, v)$ between two nodes u, v is the number of edges in the path (throughout this chapter a path means a simple path i.e. nodes and edges do not repeat) from u and v. The *centre* is a node $\gamma \in V(T)$ for which $\max_{v \in V(T)} d(\gamma, v) = \min_{u \in V(T)} \{ \max_{v \in V(T)} d(u, v) \}$. A tree T is *linkless* if it has no nodes of degree two (except possibly the centre when the radius of T is greater than one). A *2-link* tree is one that can be obtained by subdividing every edge of a linkless tree with exactly one node each. $L_2(k)$ is the class of *2-link* trees which have a radius $2k$ (see Figure 4.1).

A *caterpillar* is a tree in which there exists a path such that any node not on the path is a leaf (a node of degree one). Such a path is called a *spine* and the leaves are called the *hairs*. The notion of a caterpillar can now be generalised by defining a *caterpillar number* for every tree. For this we first define the caterpillar number of a maximal path in a tree. Given an arbitrary tree T the hairs with respect to a maximal path P in the tree are the subtrees of T *sticking out* of P i.e., the induced subtrees of T on vertex sets of the form $V(T') \cup \{v\}$ where T' is a connected component of $T - P$, $v \in V(P)$ and $\exists u \in V(T')$

60

Figure 4.3: C_1 and C_2.

(u_1, u_2) is a two-adjacent pair
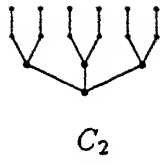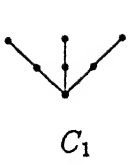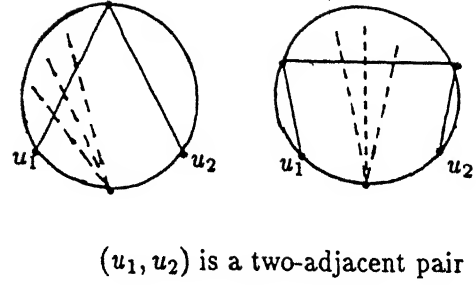
Figure 4.4: Projection constraints

such that $(u, v) \in E(T)$. Note that each hair has exactly one node in common with P. The caterpillar number of a maximal path can now be defined recursively as follows:

1. The caterpillar number of a path with respect to which every hair has exactly two nodes, is zero.

2. Caterpillar number of a path P is

$$1 + \max_{\text{over all hairs}} \left\{ \begin{array}{l} \text{min. caterpillar number in the hair over all the maxi-} \\ \text{mal paths on it starting from the node on } P \end{array} \right\}.$$

The caterpillar number of a tree is now the minimum caterpillar number over all maximal paths of the tree. Trees of caterpillar number k will henceforth be referred to as *k-caterpillars* (see Figure 4.2).

We now give a structural characterization of k-caterpillars. For every k the graph C_k defined below is the smallest possible k-caterpillar (see Figure 4.3).

1. C_1 is the tree in $L_2(1)$ with seven nodes.

2. C_k is obtained by removing a leaf each from three C_{k-1}'s and merging the three nodes from which the leaves were removed, into a single node.

An *edge-contraction* in a tree consists of removing an edge from the tree and merging the end-vertices of the edge into a single node. A tree is said to be *contractable* to another if

61

it can be transformed into the other by a sequence of edge-contractions. We now have the following easy lemma.

Lemma 4.1.1: *A tree is of caterpillar number k if and only if it contains a subtree contractable to C_k and no subtree contractable to C_{k+1}, $k \geq 0$.*

4.2 Embeddings and Meshability

This section is a brief review of the results in [36, chapter 7]. Here we consider a special embedding of trees in the plane—one in which the nodes of the tree lie on the circumference of a circle. Henceforth by an embedding we mean this kind of an embedding. An embedding of a tree is realizable if there exists a polygon realizing the tree such that the edges corrresponding to the nodes of the tree are in the same order along the boundary as that of the nodes along the embedding circle. The following are the necessary and sufficient conditions for an embedding to be realizable.

1. The embedding must be planar i.e., the images under the embedding of no two distinct edges of the tree intersect except possibly at common end-point.

2. The distance on the tree between any pair of consecutive nodes (called a *2-adjacent pair* on the embedding circle is at most three.

It is known that any tree has at least one realizable embedding. Throughout the rest of this chapter, we use the same names, both for the vertices of the trees to be meshed and their embeddings on the plane as desribed above.

When there are a pair of trees, we embed them similarly but with the nodes of the two trees alternating on the embedding circle. The correspondence between an embedding and a realizing polygon is exactly as in the single tree case. The joint realizability of two trees is now equivalent to the existence of a realizable embedding of both together. The

62

conditions necessary and sufficient for a joint-embedding to be realizable are (see Figure 4.4. Note that in all the figures showing meshings the arcs of the two trees are shown differently— typically edges of one tree, usually T_1, by solid lines and those of the other tree, usually T_2, by broken lines):

1. Both the tree embeddings individually satisfy the conditions necessary and sufficient for a single tree embedding to be realizable.

2. If the distance between the nodes in a two-adjacent pair of T_1 (respectively T_2) is two, then all the edges incident on the node of T_2 (respectively T_1) embedded between the pair project across only one of the edges of the path (on the tree) connecting the pair.

3. If the distance is three, then the edges project only across the middle edge of the path connecting the pair.

The last two are called the *projection constraints*. Henceforth whenever we refer to an embedding or a meshing we mean only a realizable embedding or meshing.

Finally we state two results which partially solve our problem.

Lemma 4.2.1: *[36] A k-caterpillar, $k \geq 1$, cannot mesh with any tree $T \in L_2(1)$.*

Lemma 4.2.2: *[36] A caterpillar can mesh with any tree having the same number of vertices.*

4.2.1 Mesh contraction

Mesh-contraction is defined on a meshing of two trees T_1 and T_2 and a pair of vertices u_1 and u_2 of T_1 such that the distance between them is at most three. The contraction is done along an arc delimited by u_1 and u_2, of the circle on which T_1 and T_1 have been embedded. For the cases where $d(u_1, u_2) > 1$, all the vertices on the path from u_1 to u_2 must lie on
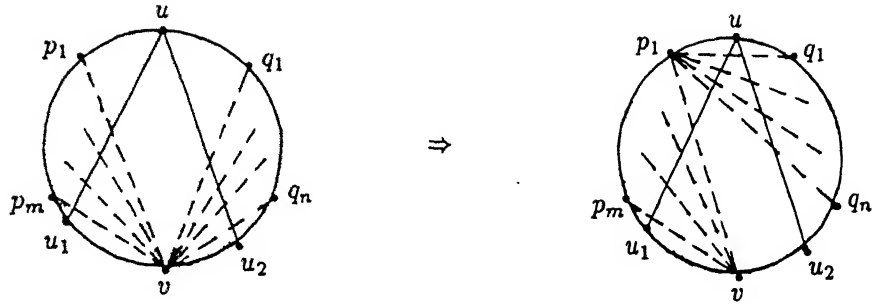
Figure 4.5: Mesh-contraction—distance two. Here p_1 and q_1 are the vertices adjacent to v that are closest to u on either side of it.

the same arc and contraction must be done along the other arc delimited by u_1 and u_2. *Mesh-Contraction* now consists of the following operations:

1. All the vertices of T_1 on the arc of contraction except u_1 and u_2 are deleted along with the edges incident on them.

2. All the vertices of T_2 on the arc-of-contraction are merged into one vertex v.

3. Multiple edges i.e., two or more edges between the same pair of vertices, incident on v are ignored.

4. If $d(u_1, u_2) = 2$ and the edges from v violate the projection constraints as shown in Figure 4.5, then every edge $\overline{vq_i}$ is replaced by $\overline{p_1 q_i}$, $1 \leq i \leq n$.

5. If $d(u_1, u_2) = 3$ and the edges from v violate the projection constraints as shown in Figure 4.6, then every edge $\overline{vp_i}(\overline{vq_j})$ is replaced by $\overline{v_1 p_i}(\overline{v_t q_j})$, $1 \leq i \leq m$ $(1 \leq j \leq n)$.

Claim: The configuration obtained after a mesh-contraction is also a meshing.

Proof: Let P be the polygon corresponding to a meshing of two trees. We now describe transformations on the polygon which mimic the operations constituting a mesh-contraction. The following cases arise:

64
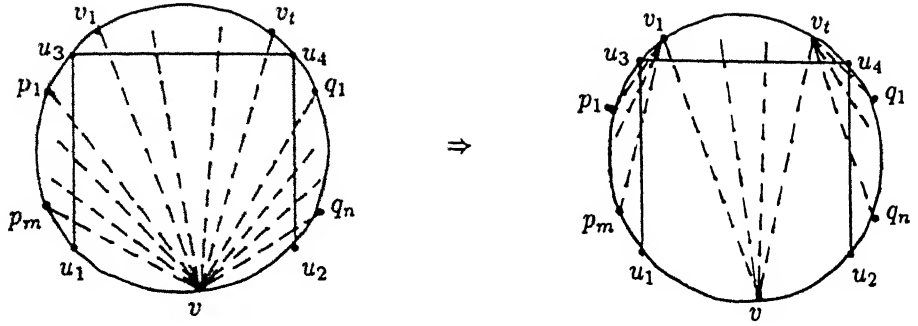
Figure 4.6: Mesh-contraction—distance three. Here v_1 and v_t are the vertices adjacent to v that are closest to u_3 and u_4 respectively in $\widehat{u_4 u_3}$.

1. $d(u_1, u_2) = 1$ (See Figure 4.7).

2. $d(u_1, u_2) = 2$ (See Figure 4.8. Figure 4.5 shows the corresponding mesh-contraction operation). The chain connecting u^* to p_1^- and that connecting u_2^- to u^* remain as they are. The chain connecting p_1^- to u_1^- is compressed vertically into the gap between u_1^- and u_2^- and p_1^- is extended below u_2^-.

3. $d(u_1, u_2) = 3$ (See Figure 4.9. Figure 4.6 shows the corresponding mesh-contraction operation). The chain connecting v_t^- to v_1^- is compressed vertically into the gap between u_1^- and u_2^- so that v_1^- and v_t^- reach below u_1^- and u_2^- respectively. The rest of the polygon remains as it is.

It is easily verified that these transformations on the polygons are equivalent to the mesh-contraction operations described earlier. The realizability of the mesh obtained after a mesh-contraction follows from the fact that the above polygon transformations lead to valid polygons. ∎

Henceforth wherever a meshing is understood we simply refer to a mesh-contraction as a contraction. From the equivalence between embeddings and polygons mentioned in

65
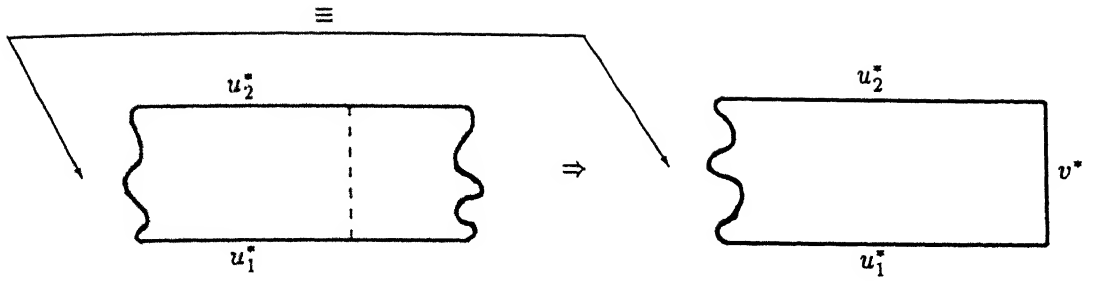
Figure 4.7: The polygon transformation for the case $d(u_1, u_2) = 1$.
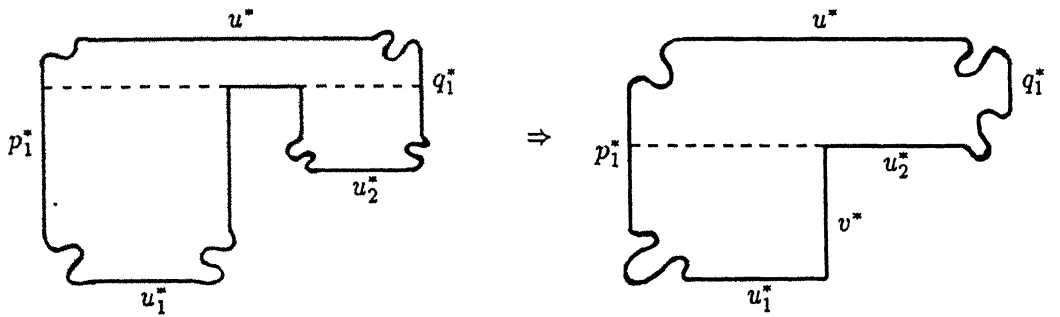
Figure 4.8: The polygon transformation for the case $d(u_1, u_2) = 2$.
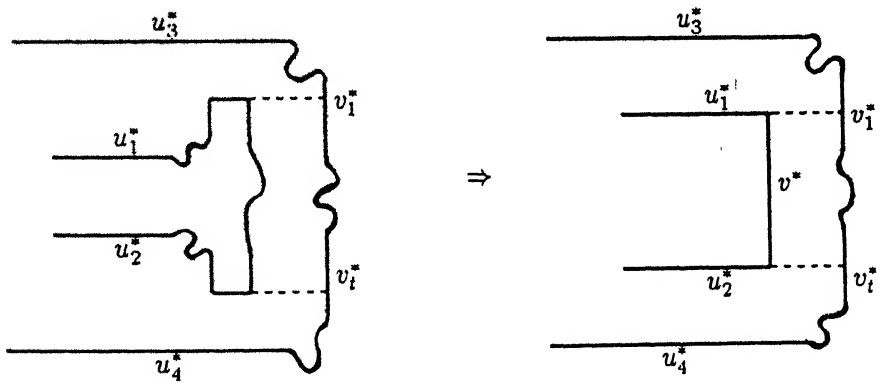
Figure 4.9: The polygon transformation for the case $d(u_1, u_2) = 3$.

the last section it follows that the question whether two trees are jointly realizable is the same as asking if the trees mesh. So in the rest of this chapter we deal exclusively with embeddings and meshings.

4.3 Caterpillars vs. 2-Link Trees

We now come to the first of our classes of invisible pairs.

Theorem 4.1: *A k-caterpillar cannot mesh with a tree in $L_2(k), t \geq k$.*

Proof: The statement is true for $k = 1$ (Lemma 4.2.2). The rest of the proof is by an induction on k. We therefore assume that the result holds for all trees in $\cup_{t<k} L_2(t)$. Consider trees T_1 of caterpillar number k and $T_2 \in L_2(k)$. T_1 has a subtree contractable to C_k. Let r be a vertex of T_1 such that at least three of the subtrees of T_1 joined at r have caterpillar number not less than $k - 1$ or at most a leaf short of being a $(k - 1)$-caterpillar. At least one such r obviously exists.

Consider an embedding of T_1. The embedding being planar, for every vertex u of T_1 it imposes a total order on the subtrees of T_1 joined at u (the order in which the vertices of the subtrees appear on the embedding circle starting from v). Let $T_{11}, T_{12}, \ldots, T_{1n}$ be the subtrees of T_1 joined at r in the order in which they are embedded. Of these at least three T_{1x}, T_{1y} and T_{1z} are of caterpillar number at least $k - 1$ or at most a leaf short of it. Each of the three sets of vertices $S_1 = \cup_{t<y} V(T_{1t})$, $S_2 = V(T_{1y})$ and $S_3 = \cup_{t>y} V(T_{1t})$ will then form contiguous arcs of the embedding circle as shown in Figure 4.10, where (a, b) and (c, d) are 2-adjacent pairs. It is straightforward to verify from the conditions necessary and sufficient for an embedding to be realizable that among a, b, c, d and r only five configurations are possible (see Figure 4.11). Note that among the three arcs $\overparen{ra}$, $\overparen{bc}$ and $\overparen{dr}$, no two vertices belonging to different arcs have an edge connecting them and that exactly one edge connects r to vertex in $\overparen{bc}$ (throughout the proof we use $\overparen{xy}$ to denote the counterclockwise arc of the embedding circle, from x to y).

67
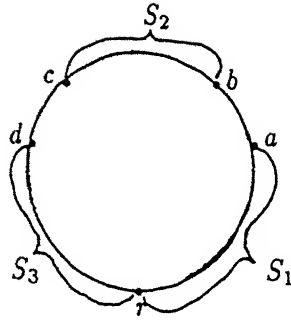
Figure 4.10: The sets of vertices S_1, S_2 and S_3.

(i)

(ii)

(iii)

(iv)

(v)
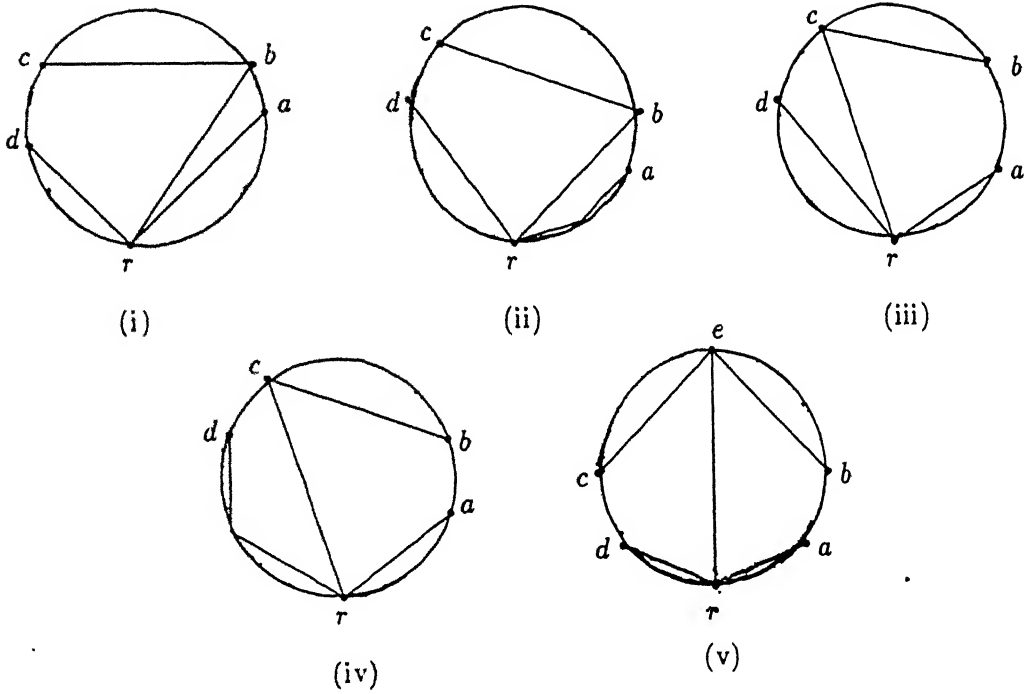
Figure 4.11: The possible generic embeddings of T_1.

68
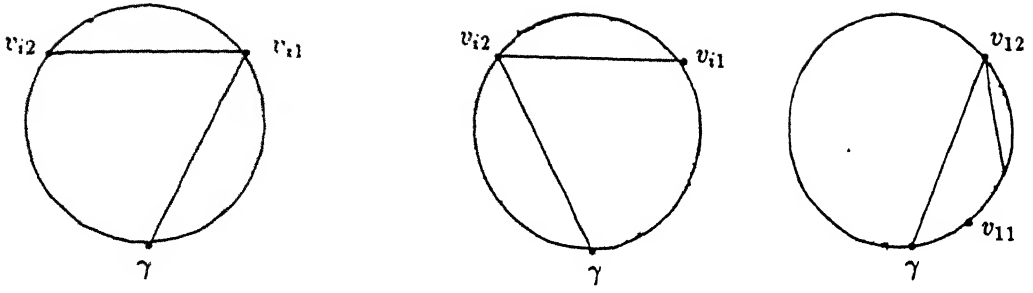
Figure 4.12: The possible generic embeddings of T_2. The first two configurations are for any T_{2i}. However T_{2m} can also have a configuration similar to the third.

In an embedding of T_2 there is a similar ordering of the subtrees $T_{21}, T_{22}, \ldots, T_{2m}$, $m \geq 2$, joined at its center γ, each of which is in some $L_2(t)$, $t < k$. In a counterclockwise traversal of the embedding circle starting from γ, for every $i \leq m$, let the first and the last vertices encountered belonging to $V(T_{2i}) - \{\gamma\}$ be called v_{i1} and v_{i2} respectively. Note that for the embedding to be realizable, for every i, γ is adjacent to at least one of v_{i1} and v_{i2} and if $1 < i < m$ then v_{i1} and v_{i2} are also adjacent to each other (see Figure 4.12).

To prove the theorem we attempt meshing T_1 with T_2 by trying to place the center γ of T_2 on the embedding circle for some arbitrary embedding of T_1. Here three major cases arise (see Figure 4.10):

1. γ lies between a and b.

2. γ lies in $\overset{\frown}{ra}$.

3. γ lies in $\overset{\frown}{bc}$.

We treat each of these cases separately below and show that in every case we are lead into a contradiction. A meshing then will not be possible as claimed in the statement of the theorem because in that case γ cannot be placed anywhere on the embedding circle.

An observation about the case analysis which follows is in order here. In many of the

cases we show a subtree of T_1 (containing at least one of T_{1x}, T_{1y} and T_{1z} as a subtree) and a T_{2i} for some $1 \le i \le m$ such that meshability of T_1 and T_2 would imply meshability of the two subtrees. To be able to invoke the induction hypothesis and arrive at a contradiction we need to ensure that the subtree of T_1 obtained is at least a $(k-1)$-caterpillar because $T_{2i} \in L_2(t)$, $t < k$. Note that because of the way the C_i's are constructed, in general a subtree of T_1 at r could be one leaf short of a $(k-1)$-caterpillar but this can happen only when r itself is a leaf in the subtree. It may be verified that in none of the cases where we invoke the induction hypothesis using a subtree of T_1 does r become a leaf. Thus in the rest of the proof we will not explicitly check for this possibility.

case 1: γ lies between a and b.

If $v_{m1} \in \widehat{br}$ then for each of the three possible generic configurations of T_{2m} as implied by Figure 4.12, a contraction along $\gamma \widehat{v_{m1}}$ would give us a meshing of T_{2m} which is in $L_2(t)$, $t \le k-1$ with a tree of caterpillar number at least $(k-1)$ (a subtree of T_1 containing T_{1x}). This violates the induction hypothesis. Therefore $v_{m1} \in \widehat{ra}$. This implies that there is at least one edge connecting γ to a vertex of T_2 in $\widehat{ra}$. Projection constraints would now be violated if T_1 is embedded the way shown in cases (iii)–(v) of Figure 4.11. Thus the only way to ensure that the cases (i)–(ii) of Figure 4.11, are also not ruled out is not to have any edge connecting γ to a vertex of T_2 in $\widehat{br}$ thus forcing v_{12} to be in $\widehat{ra}$ and to be adjacent to γ. Here again a contraction along $\widehat{v_{12}\gamma}$ leads to a meshing which violates the induction hypothesis (T_{21} with a subtree of T_1 containing T_{1y} and T_{1z}).

case 2: γ lies in $\widehat{ra}$.

Let s be the minimum integer for which $v_{s2} \in \widehat{b\gamma}$. Now if $v_{s2} \in \widehat{c\gamma}$ then irrespective of where v_{s1} is, two contractions—one along $\widehat{v_{s2}\gamma}$ and the other along $\gamma \widehat{v_{s1}}$—would lead to a mesh between a subtree of T_1 containing T_{1y} and T_{2s}. Note that because of our choice of s, the farthest v_{s1} can be from γ is just next to b counterclockwise on the circle and even in this case the contractions leave T_{1y} unaffected. Invoking the induction hypothesis again we arrive at a contradiction. Thus $v_{s2} \in \widehat{bc}$. Here if v_{s1} lies between a and b then from

the possible embeddings for a T_{2s} as shown in Figure 4.12, either there is an edge from γ to v_{s2} or there are edges from v_{s1} to both γ and v_{s2}. The second alternative will cause a violation of the projection constraints at v_{s1} for every possible configuration of T_1 shown in Figure 4.11 because the vertex between a and b (i.e., v_{s1}) projects into a vertex each in both $\overset{\frown}{ra}$ (in this case γ) and $\overset{\frown}{bc}$ (in this case v_{s2}). So an edge from γ to v_{s2} (essentially an edge from a vertex in $\overset{\frown}{ra}$ to one in $\overset{\frown}{bc}$) is the only alternative left. This will happen even when $v_{s1} \in \overset{\frown}{ra}$ or is the vertex next to b counterclockwise on the embedding circle. But an edge from a vertex in $\overset{\frown}{ra}$ to a vertex in $\overset{\frown}{bc}$ rules out configurations (iii) and (iv) of Figure 4.11 because the edge will prevent the vertex embedded between a and b from projecting across $\overline{rc}$ as the projection constraints demand. So we now need to check only for the configurations (i), (ii) and (v) of T_1 shown in Figure 4.11.

Suppose t is an integer such that v_{t1} lies between c and d. Such a t cannot exist because if $v_{12} \in \overset{\frown}{dr}$ then in all the three remaining configurations of T_1 projection constraints get violated at v_{t1} (projection across $\overline{rd}$) and if $v_{12} \in \overset{\frown}{r\gamma}$ then contractions along $\overset{\frown}{v_{t2}\gamma}$ and $\overset{\frown}{\gamma v_{t1}}$ will lead to a mesh between T_{2t} and a subtree of T_1 containing T_{1z} thus violating the induction hypothesis.

From the observations about T_{2s} and T_{2t} above it follows that there exists a largest l for which $v_{l1} \in \overset{\frown}{bc}$. Here again if $v_{l2} \in \overset{\frown}{r\gamma}$ we are lead into a contradiction to the induction hypothesis after contractions along $\overset{\frown}{v_{l2}\gamma}$ and $\overset{\frown}{\gamma v_{l1}}$. In both cases we get a mesh between T_{2l} and a subtree of T_1 containing T_{1z}. Thus $v_{l2} \in \overset{\frown}{cr}$. Since $l > 1$, in any configuration of T_{2l} as shown in Figure 4.12, $(v_{l1}, v_{l2}) \in E(T_2)$. So whether v_{l2} is between c and d or is in $\overset{\frown}{dr}$ projection constraints get violated for configurations (i) and (ii) of T_1 shown in Figure 4.11. If it is between c and d projection across $\overline{bc}$ is illegal (because $v_{l1} \in \overset{\frown}{bc}$) and if it is in $\overset{\frown}{dr}$ then there is an edge from a vertex in $\overset{\frown}{dr}$ to a vertex in $\overset{\frown}{bc}$ which would prevent the vertex embedded between c and d from projecting across $\overline{rb}$. Thus T_1 cannot be embedded as in configurations (i) and (ii) of Figure 4.11 too.

Finally we are left with only configuration (v) of Figure 4.11. We have also shown

in the analysis above that there exist edges of T_2 from $\overset{\frown}{ra}$ to $\overset{\frown}{bc}$ and also from $\overset{\frown}{dr}$ to $\overset{\frown}{bc}$. To enable the vertex embedded between c and d and that between a and b to satisfy the projection constraints, it is necesary that for configuration (v) of T_2 the edge from $\overset{\frown}{ra}$ to $\overset{\frown}{bc}$ is actually between $\overset{\frown}{ra}$ and $\overset{\frown}{cc}$ and that between $\overset{\frown}{dr}$ and $\overset{\frown}{bc}$ is actualy between $\overset{\frown}{dr}$ and $\overset{\frown}{be}$. But two such edges would cross and therefore such an embedding of T_2 would violate planarity. Since no other possibilities remain we conclude that γ cannot lie in $\overset{\frown}{ra}$ if T_1 and T_2 are to mesh.

case 3: γ lies in $\overset{\frown}{bc}$.

In this case too the general idea is to take the smallest s for which $v_{s2} \in \overset{\frown}{c\gamma}$ and the largest t for which $v_{t1} \in \overset{\frown}{ra}$. After ruling out all other possibilities we end up with an edge from $\overset{\frown}{ra}$ to $\overset{\frown}{bc}$ and another from $\overset{\frown}{dr}$ to $\overset{\frown}{bc}$. That this too is impossible follows from the last argument which settled case 2. We omit the details here because the earlier two cases have been treated exhaustively and the arguments for this case are almost identical. The reader may convince himself that here too one is always led into a contradiction.

Having treated all the three cases we conclude that γ cannot be legally embedded anywhere on the embedding circle for any arbitrary embedding of T_1. The choices of the trees and the embeddings having been totally arbitrary we conclude that no tree having C_k as a subtree can mesh with a tree in $L_2(k)$. The theorem follows from the fact that a s-caterpillar contains as subtrees all C_t's where $t < s$. ∎

The converse of Theorem 4.1 is unfortunately not true. An example of a pair of trees for which the converse fails to hold is shown in Figure 4.13, where $T_1 \in L_2(2)$ and T_2 is a 1-caterpillar. That these two do not mesh follows from the other negative result that prove in the next section.
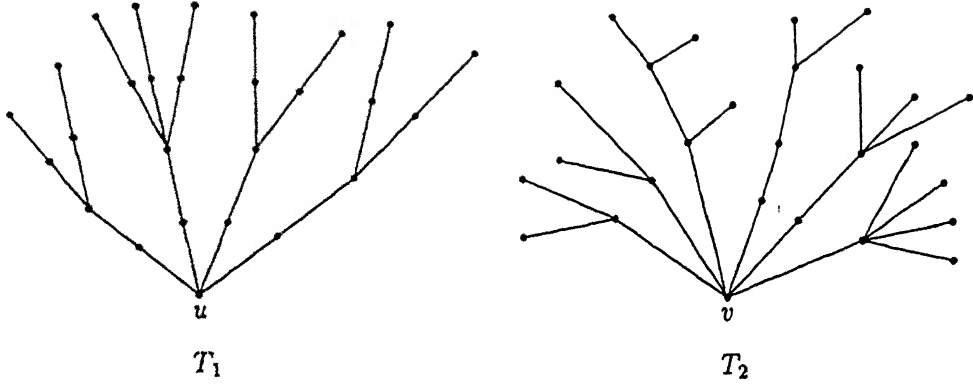
Figure 4.13: A pair of unrealizable trees not covered by Theorem 4.1.

4.4 Subtree Sizes and Unrealizability

Theorem 4.1 demonstrates a class of trees which are not jointly realizable. In the rest of this section we construct another class of such jointly unrealizable pairs of trees.

Consider a tree T_1 with a distinguished vertex u and let $T_{11}, T_{12}, \ldots, T_{1s}$, $s > 4$, be the subtrees of T_1 at u with $1 < n_i = |V(T_i)| - 1$, $1 \leq i \leq s$. Let n_δ be the fourth in a non-increasing order of the n_i's. We henceforth talk only of a typical, though general, embedding of T_1 in which the order induced on the subtrees of T_1 at u is that given above. A *boundary-pair* with respect to u is a 2-adjacent pair (u_1, u_2) such that $u_1 \in T_{1x}$ and $u_2 \in T_{1y}$, $x \neq y$. Here and in the rest of this section we refer only to boundary-pairs of T_1 with respect to u. Since $n_i > 1$, $1 \leq i \leq s$, it is easy to verify that the distance between u_1 and u_2 on the tree T_1 is three for at least $(s - 2)$ of the boundary-pairs and that for each such pair u is adjacent to exactly one of u_1 and u_2. From now on we concern ourselves only with such boundary-pairs $(d(u_1, u_2) = 3)$ and shall represent such a boundary-pair as an ordered pair (u_1, u_2) in which u_2 is adjacent to u (see Figure 4.14). Let the numbers of vertices on the embedding circle between u and u_2 (including both) and that between u_1 and w (as in Figure 4.14) be called the *armlengths* of the pair (u_1, u_2). A subtree T_{1i} is said

73
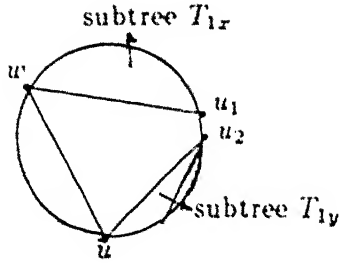
Figure 4.14: Boundary pair (u_1, u_2) with $d(u_1, u_2) = 3$, $(u_2, u) \in E(T_1)$.
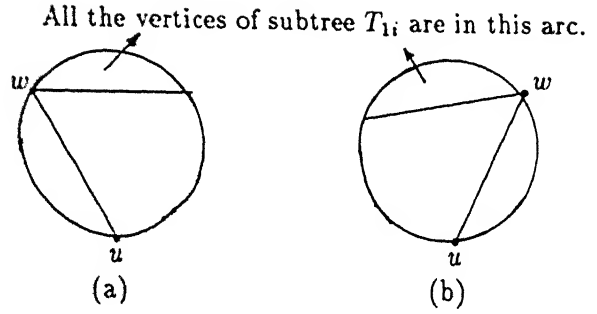
Figure 4.15: (a) clockwise and (b) counterclockwise oriented subtrees.

to be *oriented clockwise (counterclockwise)* if (see Figure 4.15):

1. One of the extreme vertices of the arc (say w) containing the vertices of T_{1i} except u is adjacent to u.

2. The clockwise (counterclockwise) arc going from w to u contains all the other vertices of T_{1i}.

We now prove a few simple lemmas before we construct our second class of unrealizable pairs.

Lemma 4.4.1: *For any contiguous sequence of similarly oriented subtrees at u, if any two of the subtrees have sizes at least α then there exists a boundary-pair within the arc containing the vertices of all the subtrees in the sequence with both its armlengths at least $\alpha - 1$.*

Proof: Without loss of generality let the subtrees of the sequence be oriented clockwise. The required boundary-pair (u_1, u_2) is the one for which u_1 belongs to the subtree which is closest to u along the direction of orientation of the subtrees in the sequence, with at least α vertices (see Figure 4.16). ∎

74

First subtree (cw from u) with size $\geq \alpha$
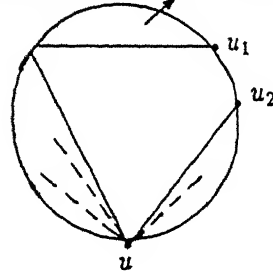
$\widehat{uu_2}$ also has a subtree
with atleast α vertices

Figure 4.16: Boundary-pair with both arm-lengths at least $(\alpha - 1)$.

separating subtree

ccw sequence   cw sequence
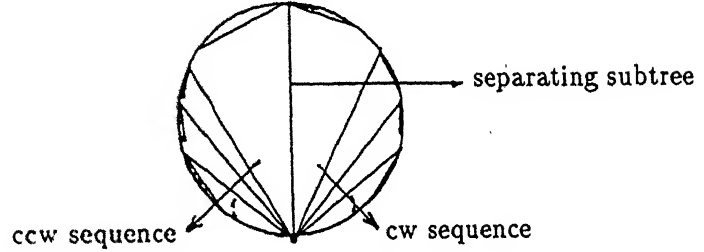
Figure 4.17: The clockwise and anticlockwise sequences of subtrees and at most one separating subtree in any embedding of T_1.

Observation 4.4.1: *For any embedding of T_1 there exist at most one maximal non-empty sequence of clockwise oriented subtrees and at most one non-empty sequence of counterclockwise oriented subtrees. Where both exist, they are separated by at most one subtree of T_1 at u. Note that an orientation for the separating subtree is not defined (see Figure 4.17).*

Lemma 4.4.2: *There exists a boundary-pair, both the arm-lengths of which are at least n_δ.*

Proof: From the definition of n_δ, there exist $n_x, n_y, n_z \geq n_\delta$. From the observation above,

75

the worst case embedding of T_1 is one with two oppositely oriented sequences of subtrees of T_1 at u with exactly one of the T_i's separating them. Even if $\theta \in \{x, y, z, \delta\}$, there would still be three subtrees i.e., $\{T_{1x}, T_{1y}, T_{1z}, T_{1\delta}\} - T_{1\theta}$ belonging to the two sequences of which at least two will belong to one sequence. The existence of the required boundary-pair now follows from Lemma 4.4.1. ∎

Consider now a tree T_2 of the same size as T_1 with a distinguished vertex v and the subtrees of T_2 at v being $T_{21}, T_{22}, \ldots, T_{2t}$ such that $2 < |V(T_{2i})| \leq n_\delta$, $1 \leq i \leq t$.

Observation 4.4.2: *T_2 does not contain any vertex v' such that at least two of the subtrees of T_2 at v' have more than n_δ vertices each.*

Proof: From the way T_2 is defined v obviously does not have any subtree connected to it with more than n_δ vertices. Consider any other vertex v' and say it belongs to some T_{2i}. Of the subtrees of T_2 at v' exactly one of the subtrees contains v and all others are subtrees of T_{2i} none of which can have more than n_δ vertices since $|V(T_{2i})| \leq n_\delta$. So at most one subtree of T_2 at v' can have more than n_δ vertices and our observation follows. ∎

Theorem 4.2: *T_1 and T_2 are not jointly realizable.*

Proof: Consider an arbitrary embedding of T_1. Lemma 4.4.2 tells us that there definitely exists a boundary-pair both of whose armlengths are at least n_δ. Let such a boundary-pair be (u_1, u_2) and let v_1 be the vertex of T_2 embedded between u_1 and u_2 (see Figure 4.18). For the projection constraints at v_1 to be satisfied, at least one of the following two conditions must be true:

1. v_1 is a leaf.

2. At least two of the subtrees of T_2 at v_1 (the first and the last in the ordering induced by the embedding on the subtrees) have more vertices than the minimum of the armlengths of (u_1, u_2).

76
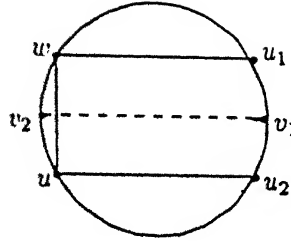
Figure 4.18: T_1 and T_2 are not jointly realizable.

Since both the armlengths of (u_1, u_2) are at least n_δ, Observation 4.4.2 implies that no vertex of T_2 can satisfy Condition 2. So the only way to mesh T_2 with T_1 is by making v_1 a leaf of T_2. Let v_2 be adjacent to v_1.

We now try to place v on the embedding circle of T_1. Since neither v nor any of the vertices adjacent to it are leaves v cannot be in either of v_1 and v_2. Suppose $v \in \widehat{v_2 v_1}$. For the embedding of T_2 to be planar, it is necessary that all its vertices in $\widehat{v_1 v_2}$ inclusive of v_1 and v_2 belong to a single subtree of T_2 at v. Since $\widehat{v_1 v_2}$ contains one arm of the boundary-pair (u_1, u_2), the number of vertices of T_2 in it including v_1 and v_2 is certainly greater than n_δ. Thus for a meshing we need a subtree at v of size more than n_δ which does not exist according to our definition of T_2. Hence $v \neq \widehat{v_2 v_1}$. An identical argument will show that neither can v be in $\widehat{v_1 v_2}$. Having thus explored all possible ways of embedding v without success we conclude that a meshing is impossible. ∎

A weak extension of Theorem 4.2 now follows easily as a corollary for the case where u also has leaves adjacent to it in T_1.

Corollary 4.4.3: *There does not exist a meshing between T_1 and T_2 if T_1 is embedded in such a way that in the ordering of the subtrees at u induced by the embedding, the number of maximal contiguous chains of non-leaf subtrees is one.*

Using this corollary as the basis, many of the restrictions imposed on the structure of

77

T_1 can now be removed by choosing an appropriate n_δ while still ensuring that Theorem 4.2 continues to hold. T_2 however is defined just the way it was earlier—that each of its subtrees at r are of size at least three and at most n_δ. We relax the restrictions on T_1 in steps through the lemmas that follow. To avoid too many repetitions, in the statement of each of the lemmas we only give the new (weaker) restrictions on T_1 and the appropriate n_δ. Every time we prove that T_1 and T_2 are jointly unrealizable. Here too we just produce a boundary-pair with both armlengths of size at least n_δ to establish joint-unrealizability. The rest of the proofs are identical to that for Theorem 4.2.

Lemma 4.4.4: *The degree of u is at least $(2l+4)$ where l is the number of leaves adjacent to u and n_δ is the $(l+4)^{th}$ in a non-increasing sequence of the n_i's.*

Proof: For an arbitrary embedding of T_1 let l' be the number of maximal contiguous chains of non-leaf subtrees at u. It is easy to see that Observation 4.4.1 holds for each of these maximal chains. Therefore at most l' of the non-leaf subtrees at u will fail to have a well defined orientation. Let $T_{1x} \prec T_{1y} \prec T_{1z}$ be three subtrees at u such that n_{1x}, n_{1y} and n_{1z} are the $(l'+1)^{th}, (l'+2)^{th}$ and $(l'+3)^{th}$ in a non-increasing sequence of the n_i's (not necessarily in that order) where "$\prec$" is the ordering imposed on the subtrees by the embedding. Note that all of these subtrees have a definite orientation. Now it is straightforward to verify that both the armlengths of the boundary-pair (u_1, u_2), $u_1 \in V(T_{1y})$ are at least n_δ' where n_δ' is the $(l'+3)^{th}$ in a non-increasing sequence of the n_i's. The lemma now follows easily from the fact that l' is at most $(l+1)$. Note that the constraint on the degree of u is because we require that n_δ is greater than one. ∎

Let $n_1' \geq n_2' \geq n_3'$ be the sizes of the three largest connected components of $T_1 - (\{u\} \cup A)$ where A is the set of all vertices adjacent to u. Intuitively these are one less than the sizes of the three largest subtrees of T_1 at *level two* with respect to u. The next lemma removes even the restriction that the degree of u is at least four more than twice the number of leaves adjacent to it by relating n_δ to n_1', n_2' and n_3'. Two of these level two subtrees

78

would be called *brothers* if they belong to the same connected component of $T_1 - u$. For convenience we would call two of the n_i''s as brothers if their respective level two subtrees are.

Lemma 4.4.5: *Let l be the number of leaves adjacent to u as defined earlier and let n_5' be the $(l+4)^{th}$ of the n_i's ($n_5' = 0$ if the degree of u is less than $(l+4)$). If $n_2' > n_5'$ then let n_5'' be the largest n_j such that $n_2' > n_j$. Now n_6 is defined as follows:*

$$n_6 = \begin{cases} \text{if } n_1' \text{ and } n_2' \text{ are brothers then} & \max(n_5'', n_3') \\ \text{elseif } n_3' > n_5' \text{ and } n_3' > 2 \text{ then} & n_3' \\ \text{elseif } n_5' > 2 \text{ then} & n_5' \\ \text{elseif } n_3' > 2 \text{ then} & n_3' \\ \text{else} & \textit{undefined} \end{cases}$$

Proof: We now prove that Theorem 4.2 would hold for n_6 defined as in each of the first four 'if'-clauses above. The order in which the clauses are to be tested for ensures that the n_6 obtained is as large as possible. Also it is easy to see that the third clause is simply a restatement of Lemma 4.4.4. We now handle the other cases below.

case 1: $n_6 = n_3'$.

Let $T_x' \prec T_y' \prec T_z'$ be the level-two subtrees of sizes n_1', n_2' and n_3' (not necessarily in that order) where "$\prec$" denotes the order which the embedding of T_1 imposes on the subtrees. Let T_y be the subtree at u containing T_y' with the vertex of T_y adjacent to u being w. It is now easy to verify that if all the vertices in $V(T_y')$ are in $\widehat{wu}$ $(\widehat{uw})$ then the boundary-pair (u_1, u_2) for which u_1 is the vertex of T_y farthest from w in $\widehat{wu}$ $(\widehat{uw})$ has both its armlengths at least n_3'. This therefore settles the second and the fourth clauses in the definition of n_6 completely.

case 2: $n_6 = \max(n_5'', n_3')$.

The proof of case 1 also in fact partially settles case 2 i.e., when $n_3' > n_5''$. Again

79

let T_y be the subtree at u containing level-two subtrees of sizes n'_1 and n'_2 with w being its vertex adjacent to u and let T_x be the subtree at u of size n''_δ. The boundary-pair (u_1, u_2) is now the one with both its armlengths at least n''_δ where u_1 is the vertex of T_y farthest from w in $\widehat{wu}$ ($\widehat{uw}$) whenever the vertices of T_x lie in $\widehat{wu}$ ($\widehat{uw}$). It may be observed that the above argument holds only when n'_1 and n'_2 are brothers. Otherwise the subtree may lie between the level-two subtrees of sizes n'_1 and n'_2 in the ordering induced by the embedding. ∎

Lemma 4.4.5 thus establishes a condition other than those implied by Theorem 4.1 which guarantees joint-unrealizability of two trees. Stated comprehensively Lemma 4.4.5 would read thus:

Theorem 4.3: *Given two trees T_1 and T_2, if there exist vertices $u \in V(T_1)$ and $v \in V(T_2)$ such that for any subtree T_{2i} of T_2 at v, $2 < |V(T_{2i})| < n_\delta(u)$, where $n_\delta(u)$ is as defined in Lemma 4.4.5 then T_1 and T_2 are jointly-unrealizable.*

Incidentally checking for the applicability of Theorem 4.3 to a pair of trees is quite easy. For any arbitrary tree the sizes of all the subtrees at a vertex of the tree (along with a selection of the kth-largest subtree size, for some k and n'_1, n'_2 and n'_3 at each vertex can be computed in linear time (linear in the number of vertices of the tree) using a simple modification of a depth first search. Having done this for each of the two trees, one can verify Theorem 4.3 trivially in constant time for every pair of vertices one from each tree. Thus $O(n^2)$ would suffice for verifying Theorem 4.3 where each tree has n vertices.

4.5   Concluding Remarks

It is easy to verify that the pair of trees shown in Figure 4.13, does fall within the class of pairs captured by Theorem 4.2. The vertices u and v of Figure 4.13, satisfy the conditions of Theorem 4.2. It is equally easy to construct a tree of caterpillar number at least one having the same number of vertices as a tree in $L_2(1)$ such that Theorem 4.3 does not

apply to this pair. It may be observed that on the other hand Theorem 4.1 implies the joint-unrealizability of any such pair of trees. Thus neither class is sufficient by itself. Also the above observations show that the two are independent, in the sense that no class is a subclass of the other. Moreover, the definitions of the two classes being so entirely different in character, it is difficult to see how a reconciliation between the two can be arrived at, to yield a complete characterization.

Chapter 5

On the Notion of Completeness for Reconstruction Algorithms on Visibility Graphs

This short chapter contains a speculative note on what we call the *Completeness* criterion for reconstruction algorithms on visibility graphs. This notion of completeness for reconstruction algorithms we feel is a highly desirable feature, though not necessary as far as just the characterization of visibility graphs is concerned. Completeness is, we feel, necessary for a deep understanding of visibility graphs in general, which is of course the ultimate goal of this study. We illustrate the notion with an algorithm which reconstructs a *horizontally convex orthogonal polygon* from its orthogonal edge visibility graph consisting of a pair of trees, one of which is a caterpillar. We also show how the algorithm is used as a 'skeleton' to arrive at an algorithm which reconstructs such a polygon realizing a *weighted* pair of trees. Our algorithm is a *complete* version of the algorithm for the same problem by O'Rourke.

In section 1 of this chapter we introduce the notion of *Completeness*. In section 2 we review briefly the algorithm of O'Rourke, which reconstructs a horizontally convex polygon

from a pair of trees, one of which is a caterpillar. We give our 'complete' version of the algorithm in section 3 and follow it with an algorithm to carry out the reconstruction from a pair of weighted trees in section 4. We end with some concluding remarks in section 5.

5.1   The Notion of Completeness

A visibility graph is a combinatorial representation of a scene. From the way visibility graphs are defined it is evident that given a scene and a particular notion of visibility, the corresponding visibility graph is unique. On the other hand in reducing the scene to a graph, quite a bit of information (mostly metric information) is lost. Thus usually, many different scenes end up having the same visibility graph. Typically a reconstruction algorithm has some nondeterminacy in it, in the sense that there would be steps in the algorithm where one is allowed to choose any one of a class of objects and each choice would lead to a different scene corresponding to the visibility graph. Such an algorithm therefore can potentially produce whole class of scenes, all of which have the same visibility graph (the given one), though the algorithm may not be capable of producing *every* possible scene with the same visibility graph. An algorithm would now be called *complete* if it can generate *any* scene with given graph as the visibility graph i.e., there exists a sequence of deterministic choices for each of the steps having some nondeterminacy, such that the algorithm produces the required scene. An algorithm complete in this sense, would we feel throw much more light on the relationship between scenes and their visibility graphs in general. Moreover, ultimately to handle scenes effectively one would certainly need a representation that captures all that is relevant to a scene. Thus, if such a representation is still going to be a *graph* some more information may have to be added (one likely way is to have a *weighted* graph with weights representing *distances*) to make the reconstruction of the scene from such a graph unique. Complete reconstruction algorithms would, as one can easily see, pave the way for algorithms which reconstruct scenes from graphs with weights (or any other non-adjacency information). In such an algorithm the *extra* information in

83

the graph would now guide the choices to be made at each of the nondeterminate steps of the complete algorithm (for the non-weighted case).

An algorithm complete in the sense described above would of course make a difference only if it is computationally tractable, i.e., both—the time to make choices at the nondeterminate steps (essentially picking any one out of many objects which satisfy a certain property) and the time to produce any given output scene are polynomials in the size of the input graph. Such an algorithm may not exist for every reconstruction problem. We may then look for some *restrictions*—in terms of output scenes etc.—which would make the problem tractable.

5.2 Reconstructing Horizontally Convex Orthogonal Polygons

A horizontally convex orthogonal polygon is one which intersects any horizontal line in a single connected segment. The horizontal edge visibility graph of a horizontally convex polygon, is a caterpillar. A caterpillar is, as one may recall from chapter 4, a tree which has a path—called the *spine*—such that every vertex of the tree not on the path is a leaf. The visibility graph of a horizontally convex orthogonal polygon thus consists of a pair of trees having the same number of vertices, one of which is a *caterpillar*. It has been shown in [36] that caterpillars are *universal trees* i.e., for every pair of trees one of which is a caterpillar, there exists an orthogonal polygon (not necessarily horizontally convex) whose visibility graph is the given pair. O'Rourke [36] has also given an algorithm to reconstruct a horizontally convex polygon from such a pair of trees. This algorithm is clearly not complete because it can produce only horizontally convex polygons. In fact it is incomplete in a stronger sense; it cannot even produce all the possible horizontally convex polygons from the given pair of trees. Here we modify the algorithm in [36] to remove the latter incompleteness.

In what follows we use the term 'polygon' to mean a 'horizontally convex polygon'. We use the labels T and C to refer to the vertical and the horizontal visiblity trees of our polygon. Note that C is a caterpillar. The rest of the notations we use here are from chapters 3 and 4.

5.2.1 Review of the known algorithm

We now give a brief description of the reconstruction algorithm in [36]. The algorithm attempts to reconstruct a polygon from a tree T and a caterpillar C. Any tree has a unique bipartition and so has a caterpillar. It would be convenient for us to refer to the partitions of C as the *left* and the *right* partitions, having l and r vertices respectively. Suppose T has an edge e which admits an embedding of T which maps e to a vertical line segment, and embeds $l-1$ vertices of T to the left of e and $r-1$ to the right. It is easily shown that then there exists a polygon with T and C as its vertical and horizontal visibility graphs respectively. A polygon obtained this way is called an *hourglass* polygon and the two trees C and T are said to *balance* each other. To obtain a joint realization of a pair of trees C and T which do not balance, a subtree C' of C is balanced with a subtree T' of T to get an hourglass polygon and the remaining vertices $C' - C$ and $T' - T$ are gathered in an isolated region. The process is repeated with the "leftover" diminishing at each step resulting in a series of hourglass polygons, horizontally displaced and connected top to bottom, which jointly realize C and T. We describe the algorithm in a little more detail below.

Let the bipartition of C be (l_1, r_1) with $l_1 + r_1 = n$ and $l_1 - r_1 = \delta \geq 0$, n being the number of vertices in C. Choose an arbitrary vertex b_1 of T as the "base". Choose an edge e_1 of T incident on b_1 such that the remaining vertices of T may be arranged on either side of e_1, with L_1 vertices to the left and R_1 to the right (note that then $L_1 + R_1 + 2 = n$), such that the quantity

$$\beta_1 = | \, (L_1 - R_1) - \delta_1 \, |$$

is minimal among all edges e_1 incident on b_1 and all arrangements of vertices. That is, L_1

85

Figure 5.1: The bipartition of a caterpillar shown as two columns (the 'columns' are shown horizontally here with 'bottom' on the right)

and R_1 are as close to l_1 and r_1 as is possible for the choice of b_1; β_1 is a measure of the "imbalance" between C and T. Clearly, if $\beta_1 = 0$ then C and T balance each other into an hourglass polygon. So let $\beta_1 > 0$ and let t_1 be the top vertex at the other end of e_1. Call the side (right or left) which has an excess (or equal amount) of T vertices over C vertices as the *high side*. The following facts can now be established:

1. t_1 must have descendents to the high side.

2. If $\triangle_1$ is the minimum among the numbers of vertices in the subtree descendents of t_1 to the high side, then, $\triangle_1 \geq \beta_1$. Thus if the subtree descendent of t_1 with $\triangle_1$ vertices is removed then the remaining number of vertices L_1' and R_1' to the left and right of e_1 are both strictly less than l_1 and r_1 respectively.

Now $\triangle_1$ vertices are removed from the high side and the remainder is balanced with a part of C into an hourglass polygon. The $\triangle_1$ vertices are then "slid" off to start another hourglass polygon. Consider a layout of the vertices of C in two columns with the vertices in each partition (of the bipartition) belonging to a column (see Figure 5.1). An edge belonging to the spine is called a *diagonal*. Let a diagonal d have d_L and d_R vertices of C below it on the left and right sides of the layout respectively. Define diagonal D to be lowest diagonal such that either $L_1' < D_L$ and $D_R < R_1'$ (see Figure 5.2a),or $L_1' > D_L$ and $D_R > R_1'$ (see Figure 5.2b); D always exists. Now if $D_L > L_1'$, then the slide will be to the left else the slide will be to the right (see Figure 5.3). The reader is referred to O'Rourke[36] for the details. It suffices to say that the procedure does converge ($\triangle_1$ strictly decreases each time) and produces the required polygon.

86
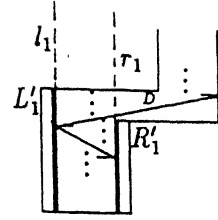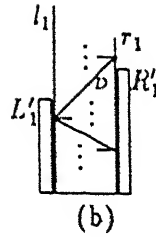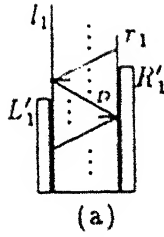
Figure 5.2: The diagonal D

Figure 5.3: A 'slide' to the right

We now make some observations on the polygon produced by the above algorithm, before we move on to our variant of this algorithm.

- The bottommost horizontal edge of the reconstructed polygon is b_1^*.

- The farthest visible horizontal edge to b_1^* is t_1^*.

- Unless t^* is the topmost horizontal edge of the reconstructed polygon P, the edge of P which can see t^* and is also in the subtree that has been slid away, is such that it can see some edge of P which is at a higher Y-coordinate than t^*. Note that there can be atmost one such horizontal edge in P.

- The vertical edges of P adjacent to b^* correspond to the end-vertices of an extreme edge of the spine of C.

5.3   The Complete Reconstruction Algorithm

Our algorithm is a variant of the algorithm described above. In the description of our algorithms, we concentrate only on the first phase of the above reconstruction scheme i.e., till a subtree descendent of t_1 is 'slid' on one side. As it can be observed, the problem 'repeats' itself after this. Thus for our algorithms, we only show how the vertices of T which would correspond to the bottommost edge b^* of P (the reconstructed polygon) and

87

the farthest visible edge t^* to b^* can be chosen to enable 'sliding' of one subtree descendent of t and balancing the remaining subtrees of t and those of b with a part of the caterpillar, into an hourglass. Wherever necessary, we will show that this can be carried out at every phase of the reconstruction, before we finally arrive at a realization of the pair of trees.

Our algorithm is based on the following observation, which is the crux of O'Rourke's algorithm: b, t and the subtree descendent g which is to be 'slid' must be such that the remaining subtrees at b and t can be arranged in such a way as to make the number of vertices embedded to the left (right) of $e = (b, t)$ in the realization, at most $l - 1$ ($r - 1$). Of course we must be able to do this in each of the subproblems we generate during the course of the algorithm. That the above condition is sufficient for reconstructing a horizontally convex polygon, is easy to see. Note that O'Rourke's algorithm uses nothing more than this property after the choice of b_1, t_1 and the subtree to be slid. Conversely, given a polygon P, the vertices of its visibility graph corresponding to its bottommost edge b^*, the farthest visible edge t^* of P to b^* and, the edge g^* which can see t^* and some other edge of P at a higher ordinate than t^*, clearly form such a set of vertices. Note that the horizontal convexity of P ensures uniqueness of g^*, if it exists. So any horizontally convex polygon can be reconstructed from its visibility trees using a scheme which chooses b, t and g as above.

Thus the following is the generic reconstruction algorithm for horizontally convex polygons. Note from our observations above, that any algorithm which reconstructs a horizontally convex polygon from a pair of trees can be rewritten to fit into this framework.

Algorithm Reconstruct

1 Choose a vertex b to correspond to the bottommost horizontal edge of the reconstructed polygon.

2 Choose a vertex t adjacent to b such that it would correspond to the vertex visible to but farthest from b in the reconstructed polygon.

3 Partition the subtrees of T at b and t, so that the partitions can be arranged on either

88

side of the edge connecting t and b in such a way that the number of vertices embedded on the left and right of the edge are less than $(l-1)$ and $(r-1)$ respectively. The subtree to be 'slid' to a side is also identified along with this.

4 Balance the caterpillar with T as far as is possible till a subtree of t is 'slid' away to a side. This starts off a new hourglass, and the above steps repeat.

End. (of Algorithmm Reconstruct)

In our algorithm, we first choose an arbitrary edge e of T in place of steps 1 and 2 of the above generic algorithm. We will eventually make one of the end-vertices of e correspond to the bottommost edge b^* of P and the other to t^*. We will now show through a simple constructive scheme, that the above assignments and the partition required in step 3 can be made for any choice of the edge e.

Lemma 5.3.1: *Let S be a finite set of positive integers and let l and r be any two positive integers such that $l + r = \sum S$, where $\sum S$ refers to the sum of the integers in S. Then there exist subsets L and R of S such that $\sum L < l$, $\sum R < r$ and $|L| + |R| + 1 = |S|$ ($|A|$ of a set A denotes the number of elements in A).*

Proof: We first give the algorithm to compute L and R and then prove its correctness.

89

Algorithm Partition

1 Initialize $L \leftarrow \phi$, $R \leftarrow S$ and $\delta \leftarrow \sum S - r$.

2 If $\delta < 0$ then **STOP**. The current L and R are the required subsets.

3 Otherwise $L \leftarrow L \cup \{i\}$ if $i \leq \delta$, $R \leftarrow R - \{i\}$ and $\delta = \delta - i$, where i is some arbitrary element of R.

Go to Step 2.

End. (of Algorithm Partition)

To prove that the above algorithm produces the required L and R, we first observe that the algorithm terminates as soon as the i in step 3 becomes greater than δ and that this can happen exactly once. Indeed if neither l nor r is zero, then irrespective of the sequence of the choices made, at some stage i must certainly sometime be greater than δ. Till this happens, the element taken out from R i.e., i, is smaller than the amount by which $\sum R$ needs to be decremented to make it less than r and thus $\sum R > r$ throughout. $\sum L$ is therefore strictly less than l. In the last step $i > \delta$ and thus when it is removed from R, $\sum R$ becomes less than r and $\sum L$ remains less than l since i does not go into L in the last step. The L and R produced by the algorithm are therefore exactly the way we want them and moreover it is easy to see that the algorithm can potentially produce any such (L, R) pair. ∎

Now let S be the set consisting of the numbers of vertices in the subtrees of T at both the end-vertices of e. Lemma 1 tells us how to compute the partitioning of S into L and R by removing one element from S. The end-vertex adjacent to the subtree corresponding to this element will be made t and the other b. The vertex of this subtree, which is adjacent to t, will then be g. Clearly, every possible legal partition for the given tree edge e can be obtained by the above scheme.

Since the partitioning algorithm makes no assumptions about the set S to be partitioned, it is clear that the reconstruction based on the above partitioning scheme, can be

carried out over all the phases of the reconstruction. Also at every stage we can potentially generate every possible partitioning, of the kind we need. We thus have a complete reconstruction algorithm for horizontally convex polygons. In the next section we use the generic algorithm to derive a reconstruction algorithm which produces such a polygon from a pair of weighted trees.

5.4   Weighted Reconstruction

Here we give an algorithm which reconstructs a horizontally convex polygon from a pair of *weighted* trees, one of which is a caterpillar. The algorithm will report a failure if a joint-weighted-realization does not exist. The interpretation of the weights we use here is that a pair of edges see each other with weight W if the distance (vertical distance if the two edges are horizontal and the horizontal distance if they are vertical) between the two edges is W. In accordance with the above interpretation, we assume that the weights assigned to the edges of the trees are positive reals. We denote the weight of an edge (x, y) of either tree as $W(x, y)$.

We will now see how each of the steps in the generic algorithm can be carried out for the weighted case.

Steps 1 and 2:   Choices of b and t.

This choice cannot now be arbitrary for a pair of weighted trees. We will now examine the choice of b.

We first observe that since b^* is the bottommost horizontal edge of the reconstructed polygon, b can certainly not be a leaf. Now consider any polygon P with b^* as the bottommost horizontal edge and t^*, the farthest visible edge to b^*. Suppose $b = v_0 v_1 v_2 \ldots v_k$ is a maximal path of the vertical visibility graph of P such that $v_1 \neq t$. Obviously $W(b, t) > W(b, v_1)$. It is also easy to see that in fact $W(v_{i-1}, v_i) > W(v_i, v_{i+1})$, $1 \le i < k$. This gives us the second condition to be satisfied by b. As for t, we know already that there

91

can be atmost one vertex g such that t is visible to it but is not the farthest such edge of P from g. Again it is easily seen that the weights on any maximal path of T starting from t but not containing either b, or g if it exists, must be such that they are strictly decreasing. Also for any edge f which can see t, $W(t,f) < W(b,t)$. Since g is the vertex whose subtree will be 'slid' out, g will serve as the 'base' for the hourglass to be started after the 'sliding'. Thus g must also satisfy properties similar to those for b. Thus there must exist a unique path $b = u_0, u_1, u_2, \ldots, u_k$ from b such that $W(u_{i-1}, u_i) > W(u_i, u_{i+1})$ whenever i is odd and $W(u_{i-1}, u_i) < W(u_i, u_{i+1})$ whenever i is even, for $i \leq 1 < k$ (we assume henceforth that k is as large as possible). This is the third condition to be satisfied by b. Note that $u_1 = t$ and the subtree of T at t, to be slid away, is the one containing u_2. One can see easily that in any weighted tree, there can be atmost two vertices satisfying the above conditions. In fact u_0 and u_k are the only vertices which satisfy this property. We can now choose either u_0 or u_k as our b. The choice does not really matter because u_0 and u_k actually become the bottommost and the topmost vertices in the reconstructed polygon. Thus the polygon obtained on taking u_0 as b will be the same as that obtained on taking u_k as b, except that the polygon obtained in one will be an 'upside-down' version of that obtained in the other.

Searching for a vertex of the kind described above is very easy. We only have to look at the weights of two edges on every path from every vertex of T. This can clearly be done in time linear in the size of T.

Finally it is also obvious that along with the choice of b we also end up identifying the vertices $t = u_1$ and $g = u_2$, both of which are also determined uniquely.

Step 3: Partitioning the subtrees at b and t and the reconstruction of P.

We start the reconstruction from b and an extreme edge of the spine of C (recall the observation we made at the end of section 2). We will see below that the reconstruction proceeds uniquely at every step once the choice of the edge of C to start with, has been made. Thus we can try a reconstruction starting from one of the extreme edges, say e_c of the spine of C. If this fails then we try with the other extreme edge. Also without loss of

generality let the leaf vertex of e_c be in the right partition of the vertices of C.

Suppose P is the polygon realizing T and C jointly, with b^* being the bottommost edge and t^* the farthest visible edge to b^*. For any two horizontal edges p^* and q^* of P such that p^* is visible to t^* and q^* to b^*, if both the edges are on the same side of LOS_{bt} (the line-of-sight between b^* and t^* as defined in chapter 3) then clearly, $W(t,p) + W(b,q)$ must be strictly less than $W(b,t)$, where $W(x,y)$ denotes the vertical distance between the edges x^* and y^*. Similarly for two edges p^* and q^*, both of which are visible to b^*, it must be that $\text{thres}(p) > W(b,q)$ or vice-versa, where $\text{thres}(x) = W(b,x) - \max\{W(x,a)|a \neq b$ and a^* is visible to $x^*\}$. Now let $\text{thres}(p)$ for a horizontal edge p^* which is visible to t^*, be defined as $W(b,t) - W(t,p)$. It is now easy to verify that the length of the smaller vertical edge adjacent to b^* is simply the minimum of $\text{thres}(p)$ over all horizontal edges p^*, other than t^* and b^*, which can see either t^* or b^*.

We now define $\text{thres}(p)$ for the vertices of T which are adjacent to either b or t. The definition is exactly the same as that in the above paragraph, with 'visibility' substituted by 'adjacency'. To start the reconstruction, we take b^* as the bottommost edge. The length of b^* is clearly the weight of e_c. Since the leaf vertex of e_c is on the right, the subtree of T containing p for which $\text{thres}(p)$ is minimum, will also be embedded on the right. The edges of C incident on the non-leaf vertex v of e_c can now be easily matched with the vertices of this subtree to get a partial realization. In this the only condition that the weights on the edges of C need to satisfy is that if two vertices x and y adjacent to v are made to correspond to counterclockwise consecutive vertical edges in that order, of the partially realized polygon, then $W(v,x) < W(v,y)$ $(W(v,x) < W(v,y))$ if the distance from b to z on T is even (odd), where z^* is the horizontal edge between x^* and y^*. Of course to make sure that the visibility between b^* and t^* is not affected, we keep track of the maximal connected portion of b^* which is currently visible from a point at $y = \infty$. The weights on the edges of C must be such that this portion is not obstructed.

To complete the algorithm, we only need to say how the construction would proceed

when the vertices on the subtrees of T that have been embedded so far are not enough to match the vertices of C for a realization. The subtree of T to be 'taken in' now for carrying out the construction is the one with the vertex p for which thres(p) is the smallest among the remaining subtrees. To see why this must necessarily be so, imagine that the base b^* has been moved up to the current height (upto which the partial realization has taken place) with the edges of the polygon obtained so far deleted. Imagine removing the corresponding vertices also from the two trees. It is now as if we start the reconstruction afresh with only the current vertex of C and b active. Clearly the 'side' of (b, t) on which a new subtree of T needs to be brought in, is the one having the leaf-vertex of the extreme edge of the truncated caterpillar. Thus the subtree to be taken in, is defined uniquely at every step of the algorithm. We now know how the reconstruction can be carried out till a subtree at t is to be 'slid' to one side. This can clearly be carried on to eventually obtain a joint-weighted-realization or report a failure. Our choices at every step having been unique, a failure anywhere will imply unrealizability of the two trees for the particular starting edge C we used. We thus have an algorithm to reconstruct a horizontally convex polygon from a pair of weighted trees.

94

Chapter 6

Stabbing Simple Planar Sets

In this chapter we present an algorithm to compute a representation of all possible stabbers of a set of simple palnar objects. Our algorithm matches the time bounds obtained by Atallah and Bajaj [3] and can handle all the objects of the kind considered by them without the restriction that the boundary of every object has a constant size storage representation. Many of the ideas which lead to our algorithm, especially that of representing the transversals in terms of *antipodal pairs*, have been explored by Rappaport [37] who has established this connection and has also given an optimal transversal finding algorithm for a set of circles in the plane.

The rest of the chapter is organized as follows. In Section 1 we will describe the primitive geometric objects we deal with in this chapter besides introducing some definitions and conventions. Section 2 contains our algorithm for computing a complete representation of the stabbing lines of a given set of objects S. In Section 3 we end the chapter by showing how this algorithm can be extended to handle some other related problems.

6.1   Geometric Preliminaries

The fundamental geometric primitives we work with are convex, smooth (continuous first derivative) finite length arcs of simple (non self-intersecting) curves which have a constant-size storage description. *We assume throughout that no pair of arcs can have more than k (a constant number) intersections.* In this chapter we work with *generalized chains* (henceforth referred to as just *chains*) of the form $x_0 e_1 x_1 e_2 x_2 \ldots e_m x_m$ in which each e_i is an arc (called an *edge* of C) with end points x_{i-1} and x_i such that $e_i \cap e_j = \phi$ whenever $j \neq i + 1 \pmod{m}$ and $e_i \cap e_{i+1} = x_i$, $1 \leq i < m$. The x_i's are called the *vertices* of C. C is called *closed* if $x_0 = x_m$ (in this case $e_m \cap e_1 = x_m$) and *open* otherwise. C is *convex* if $C \cup \overline{x_0 x_m}$ bounds a convex region of the plane, where $\overline{x_0 x_m}$ is the line segment joining x_0 and x_m.

Throughout this paper we consider only directed lines unless otherwise mentioned. For any such line L, let $\mathcal{R}(L)$ and $\mathcal{L}(L)$ denote the closed right and left half-planes of L respectively. A line L is called a *support line* (or a *tangent*) of a set of points P if $P \subset \mathcal{R}(L)$ and $L \cap P \neq \phi$. For two sets P and Q, L is said to be a *direct common tangent* between P and Q if L supports both P and Q and a *transverse common tangent* if L supports P and L^- supports Q or vice-versa, where L^- is the line which coincides with L but is directed oppositely.

Adopting the methodology introduced by Souvaine [40] in designing and analyzing algorithms for splinegons, we assume that we have at our disposal oracles which would answer certain queries on the arcs in constant time. Though the time taken by these oracles will depend on k, the maximum number of times any pair of arcs can intersect, we ignore these dependencies. The number k will not figure in our estimates of the complexity of our algorithm unless the 'order' of our problem parameter n (defined later) in the complexity is a function of k. The running times of our algorithms are expressed in terms of the times taken by these oracles to answer queries and the problem parameter n. The following is the list of constant time oracles we use. We also denote the time taken by each of these to answer a query, by a label.

96

T_{tn} Given either an angle or a point, report the tangent to the arc at the given angle or through the given point.

T_{ct} Given a pair of arcs and an angle θ, report the direct common tangent (transverse common tangent) between the two arcs whose angle is closest to and greater than θ.

A sequence $U = (u_1, \ldots, u_l)$ of integers which satisfies the following conditions:

1. $1 \leq u_i \leq m$ for every i,

2. for every $i < l$, $u_i \neq u_{i+1}$,

3. and there do not exist $t + 2$ indices $1 \leq i_1 < i_2 < \ldots < i_{t+2} \leq l$ such that

$$u_{i_1} = u_{i_3} = u_{i_5} = \ldots = a,$$

$$u_{i_2} = u_{i_4} = u_{i_6} = \ldots = b,$$

and $a \neq b$,

is called a (m, t)-*Davenport-Schinzel sequence* [13]. The maximum possible length of such a sequence is denoted as $\lambda_t(m)$. It is known [27, 43, 2] that $\lambda_t(m)$ is a very nearly linear but strictly non-linear function of m. While analysing the complexities of our algorithms we repeatedly make use of the fact that for any set of positive integers $m_1, m_2, \ldots, m_s$ such that $\sum_{i=1}^{s} m_i = m$, $\sum_{i=1}^{s} \lambda_t(m_i) \leq \lambda_t(m)$. This is easily seen from the following observation: for any s sets with the ith set S_i containing m_i symbols, a concatenation of the longest (m_i, t)-Davenport-Schinzel sequences of lengths $\lambda_t(m_i)$, $1 \leq i \leq s$, is also a legal (m, t)-Davenport-Schinzel sequence of length $\sum_{i=1}^{s} \lambda_t(m_i)$ on $\bigcup S_i$. We henceforth denote the set of all (m, t)-Davenport-Schinzel sequences as DSS(m, t).

6.2   Computing Common Transversals

In this section we develop the algorithm for obtaining a complete representation of all possible common transversals of a set of objects S, if they exist. We first describe the

class of geometric objects for which our algorithm can compute common transversals. The algorithm can handle all sets consisting of objects which are either bounded by closed convex chains or are straight-line segments. Open convex chains can also be handled by making them closed with the addition of a segment joining the end-points of the chain. It is easy to verify that a line intersects a chain if and only if it intersects the object formed as above from the chain. Finally we assume that all intersections between pairs of arcs are *proper* intersections i.e., no line through the point of intersection is a tangent to both the arcs—this includes cases where an end-point of an arc lies on another unless they share an end-point. The above assumption is merely to avoid digressions into handling of the special cases that arise than to restrict the applicability of our solution.

Henceforth we refer to both the edges (except straight edges) and the vertices of the boundary of an object jointly as the *boundary elements* of the object, or simply as the *elements*. A tangent to an element is a line touching it and supporting the object containing the element. Note that because all our objects are convex, this definition is consistent with that of the tangent to an arc. In the rest of the chapter let S refer to the set of objects for which transversals are to be computed. Also let n be the sum of the numbers of elements constituting the boundaries of the objects in S.

Consider an *angular sweep* over the range $[0, 2\pi)$. For an element u, let $L_u(\theta)$ be the tangent to u at the current sweep angle θ (we omit the angle θ whenever it is understood from the context or is irrelevant). The arcs we deal with are all smooth. Thus the set of angles at which any given edge admits tangents is an interval. The same is clearly true for the vertices of S. The angles bounding the interval for a vertex v are the angles of the tangents at v, to the arcs incident on either side of v. For the special case of a single straight line segment bounded by vertices v_1 and v_2, as an object, the angle ranges for v_1 and v_2 would respectively be $[\text{angle}(\overrightarrow{v_2v_1}), \text{angle}(\overrightarrow{v_1v_2})]$ and $[\text{angle}(\overrightarrow{v_1v_2}), \text{angle}(\overrightarrow{v_2v_1})]$. Here angle($L$) for a directed line L, denotes the counterclockwise angle which L makes with the positive X-axis. A boundary element u will be called the *lowest* at some angle during the

98

sweep if $\mathcal{R}(L_u) \subset \mathcal{R}(L_e)$ for every element $e \neq u$ that admits a tangent at that angle. We will show later that the sequence of all the lowest elements of S in the order induced by the sweep is a complete representation of the set of all the stabbers of S. We now see how this sequence can be computed efficiently.

The algorithm to compute the sequence of lowest elements mimics that proposed by Hershberger [29] for computing the lower or upper envelope of a set of segments (generalized) in the plane. We first consider only those elements of S which do not admit a tangent at $\theta = 0$. An interval $[\theta', \theta''] \subseteq [0, 2\pi]$ is assigned to every such boundary element u, θ' and θ'' being the smallest and the largest angles at which u admits a tangent. The endpoints of the intervals assigned to each of the boundary elements partition $[0, 2\pi]$ into intervals. Just as in [29], we introduce a reference point in the middle of each of these intervals. Let $\theta_1 \ldots \theta_{4n'+1}$ be the sorted order of the endpoints along with the reference points of the intervals, n' being the number of elements which do not admit a tangent at $\theta = 0$. This set of boundary elements is now partitioned into $O(\log n')$ disjoint sets with the property that the number of lowest elements in each of these sets is $O(\lambda_{k+1}(m))$ where m is the size of the set. This property is crucial to the efficiency of the algorithm because in general the number of lowest elements could be $\lambda_{k+2}(m)$. For this a binary tree is built on the θ_i's, every node i of which is assigned the angle θ_i and a set of boundary elements E_i with m_i elements in it. An element which admits tangents in the range $[\theta_a, \theta_b]$ is assigned to the set E_i associated with the lowest common ancestor i of the nodes a and b in the tree. As argued in [29], all this can be accomplished in time $O(n' \log n')$. We now establish bounds on the lengths of the sequences of lowest elements before we describe the rest of the algorithm. We may mention here that we treat multiple appearances of the same element in the lowest element sequence as if they were distinct elements.

Lemma 6.2.1: *The length of the sequence $\mathcal{E}_0$ of lowest elements of a set S with n boundary elements, a pair of which can intersect at most k times is bounded by $\lambda_{k+2}(n)$. Also the length of the corresponding sequence $\mathcal{E}_i$ for any E_i is bounded by $2\lambda_{k+1}(m_i)$.*

99

Proof: Consider a pair of lowest elements u and v. Let $e_1, e_2, \ldots, e_l$ be an alternating u-v subsequence of $\mathcal{E}_0$ such that the maximal range of angles over which e_j is the lowest is $[\theta_{j1}, \theta_{j2}]$, $\theta_{j1} < \theta_{j2}$. Clearly during the sweep from θ_{11} to θ_{l2}, both u and v admit tangents throughout the range $[\theta_{21}, \theta_{l-1,2}]$. Therefore consider the portion of the sweep from some $\theta' \in (\theta_{j1}, \theta_{j2})$ to a $\theta'' \in (\theta_{j+1,1}, \theta_{j+1,2})$ for any $1 < j < l - 1$. We know that $\mathcal{R}(L_{e_j}(\theta')) \subset \mathcal{R}(L_{e_{j+1}}(\theta'))$ and $\mathcal{R}(L_{e_{j+1}}(\theta'')) \subset \mathcal{R}(L_{e_j}(\theta''))$. Therefore there exists an angle $\theta' < \theta < \theta''$ at which $\mathcal{R}(L_{e_j}(\theta)) = \mathcal{R}(L_{e_{j+1}}(\theta))$. Thus obviously $L_{e_j}(\theta) \equiv L_{e_{j+1}}(\theta)$ is a *direct* common tangent to u and v. Therefore for every j such that $1 < j < l - 1$, there exists a direct common tangent between u and v at some angle θ lying between θ_{j2} and $\theta_{j+1,1}$. Hence clearly, l is bounded by three more than the number of direct common tangents between u and v. It can be easily verified that the number of direct common tangents is atmost the number of intersections between u and v. Thus $l \leq k + 3$. Therefore $\mathcal{E}_0 \in \mathrm{DSS}(n, k+2)$ and hence the bound on its length.

case $i > 0$: From the way the E_i's were constructed, it is clear that all the elements in E_i admit tangents at θ_i. Now imagine splitting each of the elements in E_i at these points of tangency (if the element is a vertex, the *splitting* would duplicate it). After the split let the set of those portions of the elements of E_i which admit tangents at angles less than θ_i be E_i'. Similarly let the set of those admitting tangents at angles greater than θ_i be E_i''. Also let the lowest element lists of E_i' and E_i'' be $\mathcal{E}_i'$ and $\mathcal{E}_i''$ respectively. Now if we work through the arguments similar to those in the earlier paragraph, we can see that tangents exist for both u and v in E_i' throughout the range $[\theta_{21}, \theta_{l2}]$. The length of an alternating u-v subsequence of $\mathcal{E}_i'$ thus turns out to be atmost $k + 2$. Hence $\mathcal{E}_i' \in \mathrm{DSS}(m_i, k+1)$ and analogously $\mathcal{E}_i'' \in \mathrm{DSS}(m_i, k+1)$. The bound on the length of $\mathcal{E}_i$ now follows trivially. ∎

An easy corollary of the above lemma is that if every object of S consists of just a single closed arc, then the length of $\mathcal{E}_0$ is in fact bounded by $\lambda_k(n)$. Moreover the length of the lowest element list of any subset with m elements of some E_i, $i > 0$, is also at most $2\lambda_{k+1}(m)$. Thus from the summation property of the Davenport-Schinzel bounds, which we

mentioned in Section 1, it follows that the length of the lowest element sequence for any set with m elements from the union of all the sets at the same level of the tree, is also at most $2\lambda_{k+1}(m)$. We now show how the lowest element sequence for the union of the sets at any level of the tree can be computed. Let the union at some level be called E with $|E| = m$ elements in it. The strategy is a simple *divide-and-conquer*. The set E is first partitioned into E' and E'' such that $|E'|$ and $|E''|$ differ by at most one. The lists of lowest elements of E' and E'' are computed recursively and then merged to form the list $\mathcal{E}$ for E.

Let $\mathcal{E}' = \theta_0', e_1', \theta_1', e_2', \theta_2', \ldots, e_s', \theta_s'$ and $\mathcal{E}'' = \theta_0'', e_1'', \theta_1'', e_2'', \theta_2'', \ldots, e_t'', \theta_t''$ be the sequences of lowest elements of E' and E'' respectively, where θ_i' and θ_j'' are respectively the largest angles in the range $[0, 2\pi]$ at which e_i' and e_j'' are the lowest elements of E' and E'', for $1 \leq i \leq s$ and $1 \leq j \leq t$. We also assume that the range of angles over which the elements of E' admit tangents is the same as that for E''. This is justified because if this is not true then the portions which do not have any angle in common with the other can be appended as it is, to the merged lowest element sequence. Obviously such "overhangs" can occur only at the ends and can therefore be detected and "filtered" easily during a linear scan of the two sequences. Thus $\theta_s' = \theta_t''$ and $\theta_0 = \theta_0' = \theta_0''$.

In our merge algorithm we use a and b to denote e_a' and e_b'' respectively. We use the function $\text{sub}(a, b, \theta)$ which returns a if $\mathcal{R}(L_{e_a'}(\theta)) \subset \mathcal{R}(L_{e_b''}(\theta))$, and b otherwise. The function $\text{append}(\theta, x)$ appends the angle θ and the element x, in that order, to the end of $\mathcal{E}$. The list $\mathcal{E}$ is initialized to the empty list at the start of the algorithm. The function $\text{dir-tangent}(e, f, \theta)$ reports the direct common tangent between arcs e and f that is at the smallest angle greater than θ. We assume that all the above are constant time functions. The following is a formal description of our merge algorithm.

Algorithm Merge

1 % Initialization %

$a \leftarrow b \leftarrow 1; \theta \leftarrow \theta_0;$

current-sub $\leftarrow$ sub(a, b, θ); append$(\theta,$current-sub$)$;

2 while $(\theta < \theta'_s)$ do {

current-dom $\leftarrow \{a, b\} -$current-sub;

$\theta \leftarrow \min(\theta'_a, \theta''_b,$ angle(dir-tangent(current-sub,current-dom,θ)));

% if dir-tangent(current-sub,current-dom,θ)) is undefined then it is ignored %

if $(\theta = \theta'_a)$

{ if (current-sub $= a$) append$(\theta, a + 1)$; $a \leftarrow a + 1;$}

elseif $(\theta = \theta''_b)$

{ if (current-sub $= b$) append$(\theta, b + 1)$; $b \leftarrow b + 1;$}

else

{ append$(\theta,$current-dom$)$; current-sub $\leftarrow$ current-dom;}

}

End (of Algorithm Merge)

The correctness of Algorithm Merge follows easily from the fact that the angles where the lowest of E "switches" from E' to E'' or vice-versa are the angles at which there is a direct common tangent between the elements of E' and E'' which are invloved in the switch. The algorithm therefore simply looks for the event (a change in the lowest of either E' or E'' or a switch) that would occur first and does the update accordingly.

The running time of Algorithm Merge is clearly linear in $s + t$ i.e., the sum of the lengths of $\mathcal{E}'$ and $\mathcal{E}''$, which from Lemma 6.2.1 is $O(\lambda_{k+1}(m') + \lambda_{k+1}(m''))$, m' and m'' being the number of elements in E' and E'' respectively. From the summation property of the Davenport-Schinzel bounds we mentioned in Section 1, it is clear that the merge time is $O(T_{ct}\lambda_{k+1}(m) + T_{tn})$. The constant factors are because the while loop (step 2 of Algorithm Merge) calls only the oracle to compute the common tangent and during initialization the

tangents to a and b also need to be computed separately. Now from the simple recurrence $T(m) = 2T(m/2) + O(\lambda_{k+1}(m))$ and the summation property, we find that the time taken to compute $\mathcal{E}$ is $O(T_{ct}\lambda_{k+1}(m)\log m + T_{tn}m)$. Now since the tree has $\log n'$ levels, we have $O(\log n')$ disjoint sets such that all the $O(\log n')$ lowest element sequences for these sets can be computed in total time $O(T_{ct}\lambda_{k+1}(n')\log n' + T_{tn}n')$. This again follows from the summation property. Now the elements admitting a tangent at $\theta = 0$ can also be processed just the way we had done for the E_i's, to produce their list of lowest elements. So ultimately we are left with $O(\log n)$ sets for each of which we know the lowest element sequence, with the total time spent so far being $O(T_{ct}\lambda_{k+1}(n)\log n + T_{tn}n)$.

We now go through another divide-and-conquer phase, in which we merge the $O(\log n)$ sequences obtained so far. The merging algorithm is the same as before. Now even though the final list of lowest elements is of size $O(\lambda_{k+2}(n))$ this phase takes only $O(\lambda_{k+2}(n)\log\log n)$ time which is asymptotically smaller than $O(\lambda_{k+1}(n)\log n)$ as shown in [29]. Thus we obtain an $O(T_{ct}\lambda_{k+1}(n)\log n + T_{tn}n)$ algorithm to compute $\mathcal{E}_0$. From the observations made after Lemma 6.2.1 it must be clear that this same algorithm will work in time $O(T_{ct}\lambda_k(n)\log n + T_{tn}n)$ if all objects in S are made of single closed arcs on the boundary.

We now show that $\mathcal{E}_0$ is in fact a complete representation of all the stabbers of S.

Lemma 6.2.2: *If u and v are the lowest elements of S at angles θ and $\pi + \theta$ respectively, then a line L at angle θ or $\pi + \theta$ is a stabber if and only if $L \subset \mathcal{R}(L_u(\theta)) \cap \mathcal{R}(L_v(\pi + \theta))$.*

Proof: First we observe that for any angle ψ there is an element which admits a tangent at ψ on the boundary of every object in S.

From the definition of a lowest element we have $\mathcal{R}(L_u) \subset \mathcal{R}(L_e)$ for every element e which admits a tangent at θ. Thus since one such e exists in every object of S, $\mathcal{L}(L_u)$ has a non-empty intersection with every object in S. Similarly $\mathcal{L}(L_v)$ also has a non-empty intersection with every object in S. Suppose a line L at angle θ is contained in

$\mathcal{R}(L_u) \cap \mathcal{R}(L_v)$. Then clearly $\mathcal{R}(L) \subseteq \mathcal{R}(L_u)$. Therefore $\mathcal{L}(L_u) \subseteq \mathcal{L}(L)$. It now follows that $\mathcal{L}(L)$ has a non-empty intersection with every object in S. Similarly since $\mathcal{L}(L_v(\pi + \theta)) \subseteq \mathcal{R}(L)$, $\mathcal{R}(L)$ also has a non-empty intersection with every object in S. A similar argument works for a line at angle $\theta + \pi$ as well. It is easy to see that $L \cap P \neq \phi$ for any object P and line L for which both $\mathcal{R}(L)$ and $\mathcal{L}(L)$ have non-empty intersections with P. Thus any line in $\mathcal{R}(L_u) \cap \mathcal{R}(L_v)$ is a stabbing line for S. If however $L \not\subset \mathcal{R}(L_u) \cap \mathcal{R}(L_v)$ then either $L \subset \mathcal{L}(L_u) - L_u$ or $L \subset \mathcal{L}(L_v) - L_v$. If $L \subset \mathcal{L}(L_u) - L_u$ then since P_u, the object containing u, lies wholly in $\mathcal{R}(L_u)$ we have $L \cap P_u = \phi$. Similarly if $L \subset \mathcal{L}(L_v) - L_v$ then L will not intersect the object containing v. L cannot therefore be a stabber for S. ∎

An obvious consequence of the above lemma is that S admits stabbers at angle θ or $\pi + \theta$ if and only if $\mathcal{R}(L_u(\theta)) \cap \mathcal{R}(L_v(\pi + \theta)) \neq \phi$. We call any pair of boundary elements satisfying this property at some angle θ as an *antipodal pair* at θ. Consider an angle θ snd a finite neighbourhood of θ such that

- for any angle in ψ in the neighbourhood, u and v are the lowest elements of S at ψ and $\psi + \pi$ respectvely,

- for angles greater than θ in the neighbourhood, (u, v) is an antipodal pair and,

- for angles smaller than θ, (u, v) is not or vice-versa.

It is now easy to verify that then, $L_u(\theta)$ and $L_v(\theta + \pi)$ coincide and that they form a transverse common tangent between u and v. We use this property to compute the list of antipodal pairs of S. The list is again an alternating sequence of angles and antipodal pairs. Here it is possible that two angles (not more), say θ' and θ'', occur consecutively in the list. In this case it means there do not exist antipodal pairs in the range (θ', θ''). A formal description of the algorithm is given below, the proof of whose correctness will be very similar to that for Algorithm Merge. Also clearly the algorithm works in time $O(T_{ct}\lambda_{k+2}(n) + T_{tn})$ which is asymptotically smaller than $\lambda_{k+1}(n).\log n$. Note that a pair

of arcs can admit at most two transverse common tangents and if they admit transverse common tangents then they are necessarily disjoint. Finally it is trivial to see that S admits stabbers if and only if the list of antipodal pairs is not empty.

Let $\mathcal{E}_0 = \theta_0, e_1, \theta_1, e_2, \theta_2, \ldots, e_l, \theta_l$ where e_i is the lowest element of S in the range $[\theta_{i-1}, \theta_i]$. We use a and b to denote e_a and e_b respectively. The function $\text{atp}(a, b, \theta)$ returns true or false depending on whether (a, b) is an anipodal pair at θ or not. The function trans-tangent(a, b, θ) is similar to the dir-tangent(a, b, θ) function we used in Algorithm Merge except that trans-tangent() returns the transverse common tangent. The list of antipodal pairs is initialized to the empty list at the start of the algorithm.

Algorithm Antipodal Pairs

1 % Initialization %

$\quad a \leftarrow 1; b \leftarrow$ such that the lowest element at π is e_b; $\theta \leftarrow 0$;

$\quad$ if $(\text{atp}(a, b, \theta))$ { ATP $\leftarrow$ TRUE; append$(\theta, (a, b))$;} else ATP $\leftarrow$ FALSE;

2 while $(\theta < \pi)$ do {

$\quad\quad \theta \leftarrow \min(\theta_a, \theta_b - \pi, \text{angle}(\text{trans-tangent}(a, b, \theta)))$;

$\quad\quad$ % if trans-tangent(a,b,θ)) is undefined then it is ignored %

$\quad\quad$ if $(\theta = \theta_a)$ { $a \leftarrow a + 1$; if (ATP) append$(\theta, (a, b))$;}

$\quad\quad$ elseif $(\theta = \theta_b - \pi)$ { $b \leftarrow b + 1$; if (ATP) append$(\theta, (a, b))$;}

$\quad\quad$ elseif (ATP) { ATP $\leftarrow$ FALSE; append(θ);}

$\quad\quad$ else { ATP $\leftarrow$ TRUE; append$(\theta, (a, b))$;}

$\quad$ }

End (of Algorithm Antipodal Pairs)

We have thus obtained an algorithm to compute a complete representation of all the possible stabbers of a set S in time $O(T_{ct}\lambda_{k+1}(n) \log n + T_{tn}n)$ which also works in $O(T_{ct}\lambda_k(n) \log n + T_{tn}n)$ for special cases of the problem in which all objects of S consist of single closed arcs on the boundary.

Before we conclude this section, we may mention that unbounded straight lines as objects can also be easily incorporated into the algorithm to compute the transversals. Consider any set of lines. One can partition this set into groups of parallel lines. Let R be the set of directions corresponding to groups which have two or more lines. It is easy to see that a line stabs all the lines in the set if and only if its direction is not in R. The set R for any set of m lines can be computed in $O(m \log m)$ time by a linear scan of the list of the lines in the order of their slopes. Now given a set consisting of finite objects and also lines, all that we need to do is to exclude the set R of inadmissible directions for the lines from the admissible set of transversal directions for the rest of the objects. In fact one can handle even convex unbounded chains without much extra effort. Incidentally we we say that an unbounded chain C is convex if every bounded connected subset of it is. We however still assume that the unbounded chain is made up of a finite number of primitive arcs (not necessarily of finite length). Clearly, a tangent in a given direction or at a given point on C is still defined for such chains, except possibly at some set of angles forming a contiguous real interval. This does not cause any difficulty because, C being unbounded, non-existence of a tangent at an angle simply means *absence* of any constraint on the position of a transversal to C in that direction. This also tells us that rays can also be handled just the same way.

6.3   Some Extensions

In this section we show that the above algorithm can be modified easily to yield an elegant algorithm to compute the convex hull of a set of simple convex objects, within the same time bounds. For this we replace a lowest element by a *highest element*. For a given θ this is simply the element u such that $\mathcal{R}(L_e(\theta)) \subset \mathcal{R}(L_u(\theta))$ for every other element e which admits a tangent at θ. Note that for any given angle the highest element is the one which lies on the convex hull. The rest of the algorithm is almost identical to the one for finding the stabbers with minor changes to reflect the change in the definitions.

106

Finally we mention that since stabbers exist for a set of simple objects if and only if they exist for the set consisting of the convex hulls of those objects, stabbers of a set of simple objects too can be found within the same time bound by first computing the convex hulls of the individual objects and then running the algorithm for stabbers on the resulting convex objects. Note that the size of the convex hull of a simple splinegon with m edges is $O(m)$. This is because if we treat the edges as separate objects, then two of them can intersect only at an endpoint. Thus the number of highest elements is at most $\lambda_2(m)$ which is $O(m)$.

Chapter 7

Concluding Remarks

In the first part of this thesis we have tried to further our understanding of visibility graphs in general. We have studied two classes of visibility graphs—vertex visibility graphs of simple polygons and orthogonal edge visibility graphs.

For the vertex visibility graphs of simple polygons we have proved Everett's conjecture and that gives a new necessary condition for the Ghosh's version of the visibility graph recognition problem. This condition makes it easier to think of algorithms which reconstruct a polygon to realize a given graph. It is because once blocking vertices are assigned consistently to the invisible pairs of the graph it is reasonable to expect a polygon realizing the given graph such that the set of concave vertices of the polygon corresponds exactly to the set of blocking vertices of the graph. The earlier conditions were inadequate in this respect because they never guaranteed the existence of such an assignment for any visibility graph. It is possible that consistency is the key to a complete solution to Ghosh's version of the visibility graph recognition problem. Indeed the example of Everett [19] shown in Figure 7.1 which fails to be a visibility graph with $v_1, \ldots, v_{10}$ as the chosen hamiltonian cycle in spite of satisfying all of Ghosh's necessary conditions, does not admit a consistent blocking vertex to invisible pair assignment. We show this below, thus providing an
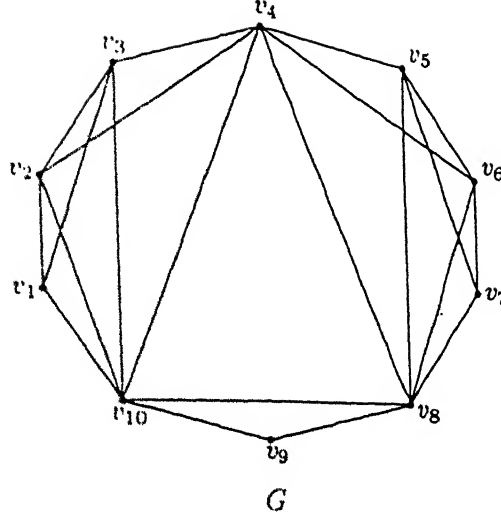
G

Figure 7.1: The graph G with the hamiltonian cycle $v_1, v_2, \ldots, v_{10}, v_1$ does not admit a consistent blocking vertex to invisible pair assignment.

alternative proof of the fact that Ghosh's necessary conditions are not sufficient.

Theorem 7.1: *The graph in Figure 7.1 does not admit a consistent blocking vertex to invisible pair assignment with respect to the hamiltonian cycle $v_1, v_2, \ldots, v_{10}$.*

Proof: Clearly v_4, v_8 and v_{10} are the only three vertices which can be blocking vertices. Note that for any other vertex v_i, v_{i-1} and v_{i+1} are adjacent, where the index arithmetic is modulo n. The pair (v_1, v_4) and (v_4, v_7) are minimal with respect to v_{10} and v_8 respectively. Now consider (v_4, v_9). Only v_{10} and v_8 are blocking vertices of this pair. If v_{10} is assigned to it then clearly (v_1, v_4) and (v_4, v_9) are separable with respect to v_{10} and the assignment is inconsistent. If however v_8 is assigned to (v_4, v_9) then there is an inconsistency with (v_4, v_7) which is also assigned v_8. Thus clearly a consistent assignment does not exist. ▌

Finally it also remains to be seen if the consistency condition can be checked for a given graph and a hamiltonian cycle in polynomial time.

The other class of graphs we have studied in this thesis is the class of orthogonal edge

visibility graphs. Firstly we have given a set of six necessary conditions for a given graph to be realizable as either the vertical or the horizontal edge visibility graph of an orthogonal polygon with holes. We have also shown that these conditions are not sufficient. However we have shown that a near-realization (upto leaves) is possible. A complete characterization however looks a far cry from here. Our algorithm to reconstruct a polygon from a given graph upto leaves works starting from any planar embedding of the graph. Even for a given planar embedding, one can start from any of the bounded regions adjacent to the outer unbounded region of the embedding. None of these need always be possible if one has to do the reconstruction using only the leaves already in the graph. Thus for a realization of the graph as it is, we have to look for an appropriate embedding of the graph in the plane. Also the order of the regions of the embedding 'added' in the construction, if indeed the construction is done that way, also has to be chosen judiciously. Since the number of possible embeddings of the graph and the number of possible ways to order the regions of the embedding are both exponential in the size of the graph, the problem is potentially an intractable one. We guess that, that is indeed the case. A related problem is to find the minimum number of leaves a given irreducible graph must have for it to be realizable.

We have also tried to make some advances towards a solution to the meshability problem for a pair of trees. In particular we have constructed two classes of pairs of trees which do not mesh. It again appears very likely that the meshability problem too in its full generality is intractable.

We have also introduced the discussed the notion of *completeness* for reconstruction algorithms on visibility graphs. We believe that algorithms complete in this sense throw a lot more light on the combinatorial nature of visibility than do algorithms which are not complete. Though the example we have chosen for illustration is a rather simple one, it seems likely that the concept will prove to be a powerful one in the study of visibility graphs.

The first part of this thesis on visibility graphs thus contributes a little towards solutions to what we believe to be very deep problems which are likely to engage researchers

for a long time to come before a fairly general, well developed 'theory of visibility graphs' emerges. However the settling of Everett's conjecture should in our opinion lead to placing the recognition problem for vertex visibility graphs of simple polygons in NP, quite soon.

We have given a very general algorithm for computing transversals of a set of objects in the plane. Besides being more general than the algorithm of Atallah and Bajaj [3] it appears far more elegant as well. Atallah and Bajaj's algorithm translates the line transversal problem into one of computing the upper envelope of functions defined on the objects of the set. These functions are by no means easy to evaluate for arbitrary convex shapes. Our algorithm on the other hand uses simple *intrinsic* functions defined on convex arcs to compute the transversals. Note that an algorithm for computing the upper/lower envelope of functions is central to our algorithm as well, but here we only 'mimic' lower/upper envelope computations instead of explicitly reducing the line transversal problem to an upper/lower envelope computation. It remains to be seen how the ideas explored here can be made use of in solving other stabbing/intersection problems.

Bibliography

[1] James Abello, Omer Egecioglu, and Krishna Kumar. Recognizing visibility graphs of staircase polygons. Manuscript 1990.

[2] P. Agarwal, Micha Sharir, and P. Shor. Sharp upper and lower bounds on the length of general Davenport-Schinzel sequences. *Journal of Combinatorial Theory, Series A*, 52(2):228-274, 1989.

[3] M. Atallah and Chandrajit Bajaj. Efficient algorithms for common transversals. *Information Processing Letters*, 25(2):87-91, 1987.

[4] David Avis, Robert J.-M., and R. Wenger. Lower bounds for line stabbing. *Information Processing Letters*, 33(2):59-62, 1989.

[5] David Avis and Godfried T. Toussaint. An optimal algorithm for determining the visibility of a polygon from an edge. *IEEE Transaction on Computers*, C-30:910-914, 1981.

[6] Chandrajit Bajaj and M. Li. On the duality of intersection and closest points. In *Proceedings of the 21st Annual Allerton Conference on Communication and Control*, pages 459-460, 1983.

[7] K. Q. Brown. *Geometric Transformations for Fast Geometric Algorithms*. PhD thesis, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, 1980.

[8] John Canny. *The Complexity of Robot Motion Planning*. PhD thesis, MIT, Boston, Massachusetts, 1988.

[9] Bernard Chazelle. Triangulating a simple polygon in linear time. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 220–230, 1990.

[10] Bernard Chazelle and Leonidas J. Guibas. Visibility and intersection problems in plane geometry. In *Proceedings of the First Annual ACM Symposium on Computational Geometry*, pages 135–146, 1985.

[11] Collette Coullard and Anna Lubiw. Distance visibility graphs. In *Proceedings of the Seventh Annual ACM Symposium on Computational Geometry*, pages 289–296, 1991.

[12] L. Danzer, B. Grunbaum, and Victor Klee. Helly's theorem and its relatives. In *Convexity, Proceedings of Symposia in Pure Mathematics, American Mathematical Monthly*, 1963.

[13] H. Davenport and A. Schinzel. A combinatorial problem connected with differential equations. *American Journal of Maths*, 87(3):684–694, 1965.

[14] M. E. Dyer. Linear time algorithms for two and three variable linear programs. *SIAM Journal on Computing*, 13(1):31–45, 1984.

[15] Herbert Edelsbrunner. Finding transversals for sets of simple geometric figures. *Theoretical Computer Science*, 35(1):55–69, 1985.

[16] Herbert Edelsbrunner and Leonidas J. Guibas. The upper envelope of piecewise linear functions: Algorithms and applications. *Discrete and Computational Geometry*, 4:311–336, 1989.

[17] Herbert Edelsbrunner, H. A. Maurer, Franco P. Preparata, A. L. Rosenberg, Emo Welzl, and D. Wood. Stabbing line segments. *BIT*, 22:274–281, 1982.

[18] H. ElGindy. *Hierarchical Decomposition of Polygons with Applications*. PhD thesis, School of Computer Science, McGill University, Canada, 1985.

[19] Hazel Everett. *Visibility Graph Recognition*. PhD thesis, Department of Computer Science, University of Toronto, 1989.

[20] Hazel Everett and D. G. Corneil. Recognizing visibility graphs of spiral polygons. *Journal of Algorithms*, 11:1–26, 1990.

[21] S. K. Ghosh and D. M. Mount. An output-sensitive algorithm for computing visibility graphs. In *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science*, pages 11–19, 1987.

[22] Subir K. Ghosh. *On Recognizing and Characterizing Visibility Graphs of Simple Polygons*, volume 318 of *Lecture Notes in Computer Science*, Springer-Verlag, 1988.

[23] J. E. Goodman and Richard Pollack. A theorem of ordered duality. *Geom. Dedicata*, 12:63–74, 1982.

[24] Leonidas J. Guibas, John Hershberger, D. Leven, Micha Sharir, and Robert Endre Tarjan. Linear-time algorithms for visibility and shortest path problems inside simple polygons. In *Proceedings of the Second Annual ACM Symposium on Computational Geometry*, pages 1–12, 1986.

[25] H. Hadwiger, H. Debrunner, and Victor Klee. *Combinatorial Geometry in the Plane*. Holt, Rinehart and Wilson, New York, 1964.

[26] Frank Harary. *Graph Theory*. Addison-Wesley Publishing Company Inc., Reading, Massachusetts, 1969.

[27] Sergiu Hart and Micha Sharir. Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes. *Combinatorica*, 6(2):151–177, 1986.

[28] John Hershberger. Finding the visibility graph of a simple polygon in time proportional to its size. In *Proceedings of the Third Annual ACM Symposium on Computational Geometry*, pages 11–20, 1987.

[29] John Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Information Processing Letters*, 33(4):169–174, 1989.

[30] M. E. Houle, H. Imai, K. Imai, and J. M. Robert. *Weighted Orthogonal Linear L_∞-approximation and Applications*, volume 382 of *Lecture Notes in Computer Science*, pages 183–191. Springer-Verlag, 1989.

[31] David G. Kirkpatrick and Stephen K. Wismath. *Weighted Visibility Graphs of Bars and Related Flow Problems*, volume 382 of *Lecture Notes in Computer Science*, pages 325–334. Springer-Verlag, 1989.

[32] F. Luccio, S. Mazzone, and C.K. Wong. A note on visibility graphs. *Discrete Mathematics*, 64:209–219, 1987.

[33] Nimrod Meggido. Linear time algorithms for linear programming in R^3 and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.

[34] Rajeev Motwani, Arvind Raghunathan, and Huzur Saran. Covering orthogonal polygons with star polygons: The perfect graph approach. In *Proceedings of the Fourth Annual ACM Symposium on Computational Geometry*, pages 211–223, 1988.

[35] Rajiv Motwani, Arvind Raghunathan, and Huzur Saran. Perfect graphs and orthogonally convex covers. *SIAM Journal on Discrete Mathematics*, 2:371–392, 1989.

[36] Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York, 1987.

[37] David Rappaport. A convex hull algorithm for discs and applications. Technical Report CISC 90-290, Department of Computing and Information Science, Queeen's University at Kingston, Canada, 1990.

[38] Micha Sharir, Richard Cole, K. Kedem, D. Leven, Richard Pollack, and S. Sifrony. Geometric applications of Davenport-Schinzel sequences. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 339–349, 1986.

[39] Xiaojun Shen and Herbert Edelsbrunner. A tight lower bound on the size of visibility graphs. Technical Report UIUCDCS-R-86-1299, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1986.

[40] Diane L. Souvaine. *Computational Geometry in a Curved World*. PhD thesis, Department of Computer Science, Princeton University, 1986.

[41] Subhash Suri and Joseph O'Rourke. Worst-case optimal algorithms for constructing visibility polygons with holes. In *Proceedings of the Second Annual ACM Symposium on Computational Geometry*, pages 14–23, 1986.

[42] Emo Welzl. Constructing the visibility graph for n line segments in $O(n^2)$ time. *Information Processing Letters*, 20:167–171, 1985.

[43] Ady Wiernik and Micha Sharir. Planar realizations of nonlinear Davenport-Schinzel sequences by segments. *Discrete and Computational Geometry*, 3:15–47, 1988.

[44] Stephen K. Wismath. Characterizing bar line-of-sight graphs. In *Proceedings of the First Annual ACM Symposium on Computational Geometry*, pages 147–152, 1985.

116577